UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Estudio y desarrollo de la automatización de una red de sensores global de honeypots

D. Eduardo Rodríguez Llorente
Tutor: Prof. D. Francisco de Borja Rodríguez Ortiz

Julio 2017

Estudio y desarrollo de la automatización de una red de sensores global de honeypots.

AUTOR: D. Eduardo Rodríguez Llorente
TUTOR: Prof. D. Francisco de Borja Rodríguez Ortiz

Grupo de la EPS 236
Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2017

Resumen

El Trabajo Fin de Grado que se presenta tiene como finalidad la detección y análisis de ataques informáticos realizados a través de Internet gracias a un sistema que se ha creado de redes de Honeypots distribuidas, conocidas como Honeynets. En total, nuestra Honeynet de investigación la componen 6 Honeypots desplegados en 4 máquinas virtuales situadas en la UAM, que servirán como prototipo para una posterior ampliación.

Primeramente, se analizarán los tipos de Honeypots que existen en la actualidad, cómo funcionan y los servicios que simulan. Paso seguido se seleccionarán aquellos que resulten más útiles para el diseño de nuestra red y se configurarán a través de pruebas de pentesting para que resulten lo más realistas posibles desde el lado de los atacantes, debido a que por defecto son fácilmente detectados.

A continuación, se integrarán los resultados obtenidos por los sensores en una base de datos única PostgreSQL, y de esta forma poder realizar consultas sobre los ataques obtenidos de una forma más eficaz. Para la visualización de dichos datos de una forma más cómoda se utilizará ELK Stack.

Una vez desplegado un prototipo del sistema de forma local y comprobado su correcto funcionamiento se procederá a realizar una clonación de los honeypots usados para analizar y estudiar el Firewall de la UAM, cambiando de posición los honeypots (delante y detrás del Firewall), y así comprobar el tipo de ataques que este bloquea.

Por último, se implementará una máquina virtual desplegable con los Honeypots seleccionados que permitirá sensorizar de forma remota a otros usuarios de una forma fácil, accesible y segura y ampliar a su vez la cantidad de datos disponibles en nuestra base de datos.

De esta forma en un futuro se llegará al objetivo de tener varios Honeypots repartidos por distintas redes, que enviarán todos los resultados obtenidos a nuestra base de datos en la UAM, donde podremos analizarlos.

Palabras clave

Honeypot, Honeynet, Glastopf, Cowrie, Dionaea, ELK stack, Kibana, Elasticsearch, Logstash, Seguridad Informática, Análisis Forense, Inyección SQL, Ataques Fuerza Bruta, Exploits, Firewall.

Abstract

This Bachelor Thesis that we introduce has as objective the detection and analysis of computer attacks made through Internet thanks to distributed Honeypot network systems known as Honeynets. In total, our investigation Honeynet is compounded by 6 Honeypots deployed in 4 virtual machines sited inside the UAM, they will be useful as prototype for later expansions.

Firstly, we will analyze the kind of Honeypots that exist nowadays, how they work and the services they simulate. Next, some of them will be selected depending of their utility to be part of our system where we will can configure them to seem like real systems using pentesting techniques, because by default tend to be detected.

Right after, the logs made from the Honeypots will be integrated in a single PostgreSQL data base providing a more efficient query system. For a comfy visualization we will use ELK stack.

Once our prototype system is deployed locally and we have checked that is working fine, we will clone it for analyze and study the effectiveness of the UAM's Firewall changing the position of the honeypots respecting the Firewall (in front of and behind the Firewall) and that way see what kind of attacks this one blocks.

Finally, an installable will be implemented with the Honeypots that we have been testing and this installable will allow to other users monitorize other networks in an easy, user-friendly way and at the same time collect all the data that they got on our data base.

This way we will reach in a future the objective of having Honeynets distributed in different networks that will send the data collected to our data base in the UAM where will we can analyze them.

Keywords

Honeypot, Honeynet, Glastopf, Cowrie, Dionaea, ELK stack, Kibana, Elasticsearch, Logstash, Computer Security, Forensic Analysis, SQL Injection, Brute Force Attacks, Exploits, Firewall.

Agradecimientos

Quisiera agradecer a todas las personas que me han estado apoyando durante estos años de carrera y han hecho posible la realización de este proyecto.

En especial, a mi familia, que estuvo allí en los buenos y malos momentos y confiaron en mí durante todos estos años. Para mí, han sido siempre un ejemplo de superación y motivación.

También le agradezco a mi tutor, D. Francisco de Borja Rodríguez, por brindarme la oportunidad de realizar este trabajo y ayudarme y asesorarme durante todo el tiempo que ha llevado su realización.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Contexto

El constante crecimiento del número de dispositivos que se conectan a internet y que estos también sean cada vez más potentes permite que también prolifere el número de ataques que se realizan a través de la red de redes. En el pasado año 2016, se registraron el doble de ataques que en el año anterior (2015) [1] en el territorio español, y posiblemente aumentarán en 2017 como está pasando en el resto del mundo [2].

Para protegernos de estos ataques existen dos estrategias: desarrollar seguridad basada en la anticipación o aplicar contramedidas sobre ataques ya registrados.

En el primer caso se utilizan estrategias comunes y otras innovadoras para atacar nuestro propio sistema y encontrar agujeros para posteriormente taparlos. La práctica más común son las auditorías de seguridad. Para sistemas SCADA[3], u otras infraestructuras críticas se deben desarrollar pruebas específicas e intentar escudarlos de la forma más hermética posible frente a futuros ataques. En España el CNPIC [4] es el órgano que se encarga de impulsar, coordinar y supervisar todas las actividades que tiene encomendadas la Secretaría de Estado de Seguridad del Ministerio del Interior, en relación con la protección de las infraestructuras críticas españolas.

La segunda opción está más enfocada al público en general, bastante menos costosa y orientada a prevenir ataques que ya hayan sufrido otros usuarios o servicios. Algunas medidas de seguridad pertenecientes a esta categoría son los antivirus y los firewalls que contienen listas de malware que se actualizan a través de una base de datos que mantiene el proveedor del servicio.

Por otro lado, los honeypots simulan un servicio web con fallos de seguridad preestablecidos, de forma que al atacante no le resulte demasiado difícil explotar las vulnerabilidades. Durante el proceso de ataque el propio honeypot registra el modus operandi del atacante. Por tanto, el honeypot cumple 2 funciones: desviar los ataques del sistema real a uno simulado y registrar dichos ataques para la posterior defensa.

Al uso de varios honeypots en una sola red se le denomina *Honeynet* y permite simular varios servicios heterogéneos con distintos protocolos como puede ser un servicio FTP, HTTP, SSH, etc. En el caso de que se registrase un nuevo tipo de ataque (zero-day), se podría analizar y estudiar para así poder proteger al sistema real frente a este, sin tener que haber sufrido dicho ataque en nuestro sistema real.

Los honeypots actuales registran los ataques y los añaden a una base de datos local por defecto. La idea principal del proyecto es la creación de una base de datos unificada, que recoja los registros creados por los honeypots independientes que estén ejecutando los distintos usuarios, y así crear una red de sensores global.

1.2 Motivación

El principal motivo para llevar acabo la realización de este proyecto es el fomentar el uso de esta tecnología tan desconocida para los profesionales de la ingeniaría informática y a la vez crear una base de datos común para todos los usuarios que quieran unirse a nuestra red.

De esta forma podremos recopilar muchos más registros sobre ataques informáticos de los que se obtendrían con un único honeypot. Así mismo, podremos además mejorar la configuración de los nuestros honeypots conociendo la configuración de los honeypots que usan los usuarios de nuestra red.

1.3 Objetivos

El principal objetivo de este proyecto es crear una honeynet global con una base de datos unificada y dar la posibilidad a otros usuarios de participar en ella. Se establecen los siguientes objetivos y en el siguiente orden:

I. Estudio de los honeypots actuales para la incorporación a la honeynet:
Análisis de los honeypots que existen para comprobar la compatibilidad que pueden tener usando una misma dirección IP, y posteriormente instalarlos en máquinas virtuales para que empiecen a recibir ataques.

II. Pentesting (o simulación de intrusión) en los honeypots para comprobar sus capacidades:
Se procederá a probar con herramientas enfocadas a auditorías de seguridad para comprobar cómo generan los registros los distintos honeypots. El tipo de auditoría se adecúa al de caja negra. Primeramente, se analizarán en los niveles de penetración de Footprinting y Fingerprinting. Estos dos primeros niveles aportarán información sobre cómo se ve el honeypot desde fuera.
Posteriormente se analizarán el nivel de análisis de vulnerabilidades y nivel de ganancia acceso y escalamiento de privilegios. Estos dos niveles finales servirán para comprobar el correcto funcionamiento del honeypot y de cómo captura los ataques.

III. Securización y ocultación de los honeypots:
Una vez analizados los honeypots procederemos a ocultarlos a los atacantes para que no sospechen y crean que están atacando un sistema real.

IV. Análisis de los ataques:
Realizadas las mejoras en los honeypots, ver los ataques que reciben, para normalizarlos y almacenarlos en una base de datos PostgreSQL.
En este caso se generarán estadísticas y se aplicará informática forense para analizar los tipos de ataques y malware.

V. Visualización de los datos de una forma rápida y cómoda:
Para comprobar el estado de las capturas de los honeypots, se implementará un visor web que muestre los datos recogidos de una forma rápida y cómoda.

VI. Validación de la Honeynet a través del Firewall de la UAM:

Se usará el Firewall de la UAM para validar nuestro sistema y comprobar que se obtienen los resultados esperados.

VII. Desarrollo de una máquina virtual distribuible para la creación de nuevos sensores y anexión a nuestra red:

La fase final constará de la creación de un archivo ejecutable que instalará una máquina virtual en el terminal del usuario que empezará a sensorizar registrando los datos, tanto localmente en la máquina del usuario como remotamente en nuestra base de datos centralizada.

1.4 Organización de la memoria y etapas del proyecto

La organización de esta memoria consta de los siguientes apartados:

- **Introducción**. Se presenta el proyecto, la motivación de este y los objetivos que se cumplirán durante su desarrollo.
- **Estado del arte**. Estudio de los honeypots actuales, cómo operan y sus principales diferencias.
- **Análisis y Diseño**. Planificación de las de las diversas etapas que forman el proyecto.
- **Desarrollo**. Contiene los procedimientos realizados para el cumplimiento de los objetivos propuestos.
- **Integración, pruebas y resultados**. Abarca tanto los resultados obtenidos como las pruebas realizadas. En nuestro caso contendrá la fase de pentesting, la securización de los honeypots y el análisis de los ataques.
- **Conclusiones y trabajo futuro**. Contiene las conclusiones finales sobre el trabajo realizado y los posibles caminos que se pueden seguir para continuar su desarrollo y evolución.

2 Estado del arte

En este capítulo detallaremos lo que es un honeypot, su funcionamiento, los distintos tipos de honeypots que existen y cómo podemos posicionarlos dentro de nuestra red.
A continuación, mostraremos las diferencias que existen con otras herramientas de seguridad informática para la prevención y defensa de ataques informáticos como pueden ser los IDS[1], SIEM[2] o Firewalls.

Introduciremos finalmente algunos de los honeypots disponibles en código abierto, que pertenecerán posteriormente a nuestra honeynet local.

2.1 Definición de Honeypot

Los honeypots (del inglés "tarro de miel") son herramientas flexibles para la seguridad informática cuyo objetivo es hacer de cebo para los atacantes maliciosos, de forma que estos crean que están atacando un sistema real, cuando en realidad están siendo monitorizados y analizado en un sistema completamente hermético y seguro para el resto del sistema.

El concepto de honeypot fue por primera vez usado durante la Guerra Fría como técnica de espionaje, pero no fue hasta 1990 cuando se trasladó el concepto al ámbito de la seguridad informática [5] en libro "The Cuckoo's Egg" de Clifford Stoll, y en el artículo "An Evening with Berferd in which a Cracker is Lured, Endured and Studied" de Bill Cheswick [6]. En la actualidad, el mayor contribuyente a la investigación con honeypots es HoneyNet Project [7][8]. Fundada en 1999, es una organización sin ánimo de lucro cuyos objetivos son la investigación, el concienciar de los peligros de los ataques informáticos, y proveer de herramientas que ayuden a promover el estudio y desarrollo de la seguridad informática.

Si bien es cierto que los honeypots pueden desviar la atención de los atacantes del sistema real, el valor de los honeypots no reside en la mitigación de dichos ataques, si no en la capacidad de aprendizaje y estudio de estos, de forma que se obtengan las metodologías y patrones que usan los atacantes en la actualidad. El problema reside en el hecho de que, aunque el atacante se encuentre dentro del sistema "falso" que genera el honeypot, puede que advierta que se encuentra en un entorno simulado e intente acceder al sistema real. Por ello es importante configurar los honeypots para que parezcan entornos reales.

Otra función que pueden cumplir los honeypots es la disuasión: configurar el honeypot para que no tenga tantas vulnerabilidades y dejar que el atacante se ofusque intentando acceder a un sistema fuertemente blindado, en el que si llega a entrar únicamente encontrará información falsa.

[1] IDS: *Intrusion Detection System*.

[2] SIEM: *Security Information and Event Management*.

La última característica a destacar es la capacidad de monitorización en tiempo real que contienen algunos honeypots que permiten detectar intrusiones. Su estrategia es similar a cómo funcionan los sistemas de detección de intrusos (IDS en inglés), permitiéndonos echar del sistema al atacante. En los siguientes apartados se mostrará más detalladamente las diferencias.

2.2 Características de los honeypots

En este apartado mostraremos un modelo genérico[5] de los elementos que componen un honeypot. Por supuesto, algunas de estas unidades son opcionales.

Principales características de los honeypots:

- Honeypot Production System: contiene la "miel" del honeypot, es decir, los datos y el sistema falso que ve el atacante y por los que creerá que es interesante atacar dicho sistema.
- Firewall: genera registros sobre las acciones que realiza el atacante para intentar introducirse dentro del sistema del honeypot.
- Monitoring Unit: monitoriza todas las actividades que realizan los atacantes dentro del sistema con todos detalles, la secuencia de comandos, los timestamps, los ficheros que ha cambiado, etc. Un IDS puede cumplir también la función de la unidad de monitorización.
- Alert Unit: genera alertas que se envían al administrador del honeypot mientras hay un ataque en ese momento.
- Logging Unit: almacena todos los registros generados por el Firewall y la Monitoring Unit.

2.3 Honeypots vs IDS y Firewalls

En esta sección se introducirán los sistemas de detección de intrusiones (IDS) y los Firewalls y se mostrarán las diferencias que poseen los honeypots respecto a estos sistemas de seguridad.

2.3.1 Firewalls

Un Firewall [9] es un dispositivo de seguridad de la red que monitoriza el tráfico entrante y saliente y decide si debe permitir o bloquear un tráfico específico en función de un conjunto de restricciones de seguridad ya definidas.

Los firewalls han sido la primera línea de defensa en seguridad de la red durante más de 25 años. Establecen una barrera entre las redes internas seguras, controladas y fiables, y las redes externas poco fiables como Internet. Un firewall puede ser hardware, software o ambos.

El uso de firewalls colocados en el borde de la red de nuestro sistema, permite controlar el flujo de paquetes entre la red de nuestro sistema e Internet. A través de información como

las direcciones IP, el nombre de los usuarios o el tipo de servicio que se solicita el Firewall puede tomar la decisión de bloquear la comunicación con un usuario externo.
Los Firewalls poseen ciertas carencias reconocidas:

- Un Firewall no protege de ataques que lo sobrepasen.
- No protege de ataques generados dentro de la propia red del sistema.
- No protege de virus o malware que vayan adjuntos en archivos y programas.
- Puede ocurrir que se produzca una sobrecarga de tráfico y el Firewall ceda en su capacidad de monitorización, permitiendo que se pueda pasar tráfico malicioso.

2.3.2 Intrusion Detection Systems (IDS)

Un IDS es un sistema que detecta y alerta sobre posibles comportamientos maliciosos dentro de la red. Se suele colocar dentro de varios puntos de la red. Suelen funcionar basados en firmas: busca firmas (o patrones) que coincidan con las firmas que tiene almacenadas en su base de datos y que son consideradas maliciosas. Si alerta al Firewall (IDS activo) de la red, puede cortar la conexión del atacante y evitar que continúe realizando su ataque, pero también contiene una serie de carencias:

- Las redes que implementan IDS deben controlar todo el tráfico que pasa por esta. Si usa un switch, un sniffer no puede obtener todo el tráfico de la red. La mayoría de las veces se suele poner el IDS en el Gateway de su conexión a Internet. Esto sin embargo impide evitar ataques internos.
- La velocidad de las redes actuales hace que un IDS tenga dificultades para ser completamente eficaz.
- Generan demasiada sobrecarga de alertas, lo que produce que cueste discernir sobre ataques importantes de falsos positivos, sobre todo en el caso de los NIDS, que son aquellos que monitorizan todo el tráfico de la red (tanto de entrada como de salida).
- El otro tipo de IDS, los HIDS, se colocan solamente en uno de los hosts y alertarán únicamente del tráfico que ese host envíe o reciba.
- Es difícil que un IDS detecte nuevos tipos de ataque si previamente no se ha registrado en su base de datos como una regla, lo que provoca que deje vulnerable a la red frente nuevos ataques.
- El hardware de los IDS debe escalar junto con el tamaño y velocidad de su red de forma que no genere retrasos significativos debido a su monitorización.
- En el caso de que las comunicaciones vayan encriptadas (porque agrega mayor seguridad obviamente), deja inutilizado el NIDS, pues no puede acceder a dicha información y compararlo con sus reglas.

2.3.3 Honeypots

Ahora mostraremos los aspectos en que mejoran los honeypots respecto a los IDS y Firewalls:

- Sólo recogen datos cuando alguien o algo interactúa con ellos lo que reduce el número de alertas comparado con los IDS. Esto hace que su análisis sea bastante más sencillo y eficiente.

- Generan un número mínimo de falsos positivos, pues introducirse dentro del sistema falso del honeypot está considerado como ilegítimo por defecto. No hay interés en los datos falsos para un usuario legítimo.
- Capturan e identifican los nuevos ataques que se hacen contra ellos, lo que nos permite estudiarlos.
- Requieren muy pocos recursos para monitorizar millones de direcciones IP.
- Aunque el ataque sea encriptado, el honeypot lo capturará.

2.4 Clasificación de los honeypots

Los honeypots se pueden clasificar en varias categorías: pueden ser clasificados según su nivel de interacción (baja, media, alta), el rol que desempeñan (servidor, cliente), el propósito de su implementación y el tipo de hardware donde se desplieguen [5, p. 31], [10, p. 25], [11, p. 23], [12, p. 19], [13, p. 32], [14, p. 28].

2.4.1 Clasificación según el nivel de interacción

2.4.1.1 Honeypots de baja interacción

Se caracterizan por tener una interacción mínima del atacante. Son los más simples y los más sencillos de mantener. Se centran en simular los servicios más usuales o susceptibles a ser atacados. Se sitúan una capa por encima del sistema operativo, lo que impide que el atacante acceda al sistema. El mayor daño que podría realizar el atacante es apagar la simulación del honeypot, nunca llegará a tomar el control del sistema.

La principal desventaja de estos honeypots es la escasez de opciones. Puede ser que no se disponga de todos los comandos de uso general que posee un sistema operativo, haciendo que el atacante se percate de que está en un honeypot y cese su actividad.
Son eficaces a la hora de identificar las direcciones IP de los atacantes, los intentos de login realizados o los servicios atacados. La mayoría de los honeypots que se usarán en este proyecto pertenecen a esta categoría.

2.4.1.2 Honeypots de alta interacción

Son sistemas convencionales con la única función de ser atacados. Son los más difíciles detectar pues lo único que es falso son los datos que se encuentren en el honeypot. Permiten obtener una huella completa sobre las actividades del atacante: desde el intento ilegítimo de login hasta la explotación de vulnerabilidades y el escalamiento de privilegios.

En este caso, es recomendable situar al honeypot en una DMZ o con un Firewall entre el sistema de producción real y este para que el atacante no pueda obtener información del resto del sistema. Estos honeypots tienen la ventaja de poder detectar ataques zero-day, aunque si no se configuran adecuadamente, el ataque o infección puede propagarse y afectar a otros sistemas de la red. Por eso, estos honeypots tienen un mayor grado de dificultad en su instalación, ya que hay que establecer unas medidas mínimas de seguridad.

Son más costos de mantener y más inseguros que el resto. Por ello se descartaron para este proyecto.

2.4.1.3 Honeypots de media interacción

Llamados también honeypots híbridos, se encuentran en un punto medio entre los honeypots de baja interacción y alta interacción. No tienen un sistema operativo real o implementan todos los detalles de los protocolos. Al igual que los de baja interacción tienen una capa de virtualización por encima del sistema operativo, lo que le brinda una mayor seguridad comparados con los de alta interacción.

Podrían considerarse honeypots de baja interacción avanzados, permitiendo mayores opciones de configuración que los de baja interacción, pero requiriendo un mayor esfuerzo para desplegarlos y configurarlos correctamente. Según las peticiones que reciba del atacante devolverá una respuesta para que el intruso se mantenga dentro del honeypot.

La línea que diferencia los honeypots de media interacción de los de baja interacción es bastante subjetiva y es por eso que algunos de los honeypots que hemos utilizado se consideren también de media interacción.

2.4.1.4 Comparación según interacción

A continuación, en la Tabla 2-1 se muestran las principales características de cada tipo de honeypot según el nivel de interacción con el atacante[5, p. 33] y cuando debería usarse cada tipo de honeypot:

Factores	Baja Interacción	Media Interacción	Alta Interacción
Grado de implicación	Bajo	Medio	Alto
Sistema operativo real	No	No	Sí
Instalación	Fácil	Difícil	Más difícil
Mantenimiento	Fácil	Fácil	Consume mucho tiempo
Riesgo	Bajo	Medio	Alto
Necesita control	No	No	Sí
Conocimiento necesario para ejecutarlo	Bajo	Bajo	Alto
Conocimiento necesario para desarrollarlo	Bajo	Alto	Medio-Alto
Recolección de datos	Limitada	Media	Extensiva
Interacción	Servicios emulados	Peticiones	Control completo

Tabla 2-1: Honeypots de baja, media y alta interacción

Viendo la tabla Tabla 2-1 podemos decidir qué tipo de honeypot usaremos en cada ocasión de forma óptima:

- Baja interacción: lo usaremos cuando no tengamos un hardware muy potente y el riesgo de usar otro tipo honeypot no es aceptable. Su objetivo será identificar escaneos y ataques automatizados además de distraer a los atacantes del sistema real.
- Media interacción: se puede asumir el pequeño riesgo que implica desplegar el honeypot en nuestra red. El principal motivo para desplegarlo será el de captar a atacantes más ávidos para que utilicen herramientas más avanzadas, además de distraerlos de acceder a los archivos importantes.
- Alta interacción: el objetivo principal será el de observar el comportamiento real de los atacantes dentro del sistema y que ayude a la práctica del análisis forense.

2.4.2 Clasificación según su rol

2.4.2.1 Honeypots modo cliente

El honeypot se convierte en un elemento activo, buscando en servidores de los que se sospeche que envían malware a los clientes que se conectan. Por ejemplo, accediendo desde un navegador web a través del honeypot para ver si el servidor intenta enviarnos algún fichero sospechoso o modificar/instalar algún plugin no deseado en nuestro navegador. También puede ocurrir con otros protocolos aparte de HTTP como pueden ser FTP o el correo.

En la Figura 2-1 se muestra el funcionamiento del honeypot en modo cliente[13, p. 29]:

Figura 2-1: Honeypot como cliente

2.4.2.2 Honeypots modo servidor

En este caso el honeypot actúa de forma pasiva simulando ser un servidor para obtener las metodologías que usan los atacantes que intentan acceder de forma maliciosa a los servidores. Un honeypot en modo servidor permite detectar nuevos exploits y malware simulando ser, por ejemplo, un servidor web o FTP; la mayoría de los honeypots de baja interacción funcionan en modo servidor. La Figura 2-2 muestra el honeypot en modo servidor.

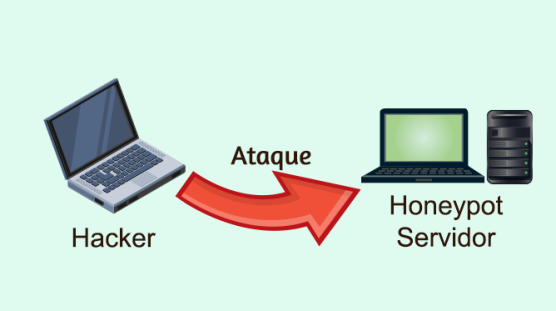


Figura 2-2: Honeypot como servidor

2.4.2.3 Otras variedades de honeypots

Honeytokens: técnicamente no es un honeypot. Consiste en marcar ciertos recursos, como documentos, mails o un registro en una base de datos, de tal forma que el acceso a estos se considere siempre como ilegítimo y, por tanto, el interactuar con ellos sea siempre interpretado como un acto malicioso.

Honeypages: son páginas web dentro de un dominio que ningún usuario legítimo alcanzaría a ver porque en el resto de la web no se les hace referencia y no están indexadas. Luego la única manera de que alguien acceda a estas páginas es a través de un escaneo intentando obtener el código fuente. Cada vez que se accede a una de estas páginas, se informa al administrador sobre el atacante.

Honeynets: al conjunto de dos o más honeypots desplegados en una misma red se le denomina honeynet. El concepto de honeynet fue propuesto por Lance Spitzner, fundador de "The Honeynet Project", en 1999 en su artículo "To Build a Honeypot"[15][16]. Al desplegar en nuestro proyecto varios honeypots en la misma red estamos creando ya de por sí una honeynet en la UAM.

Honeyfarms: en lugar de desplegar múltiples honeypots se despliegan algunos en un lugar consolidado y después se redirige el tráfico de atacantes de otras redes a la granja de honeypots[17], como muestra la imagen de la Figura 2-3:

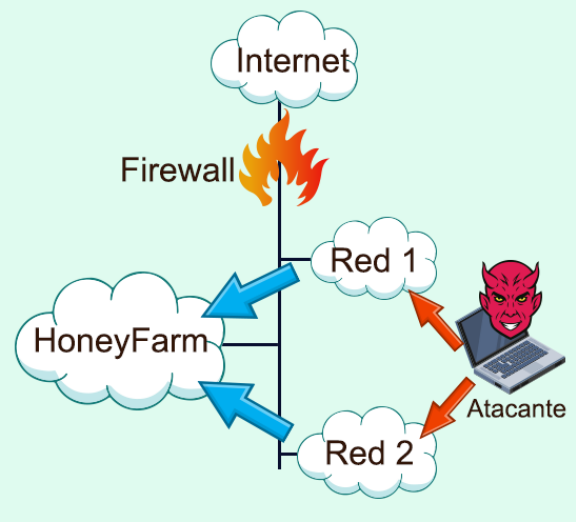


Figura 2-3: Honeypot farm

El principal problema de este método es que crea un único punto de ruptura (Single Point of Failure), de forma que, si se detecta que es un honeypot, el atacante ya lo sabrá para el futuro y si se le redirige desde otras redes no volverá a entrar al honeypot. Para el proyecto se descartó este modelo ya que preferíamos tener distintas configuraciones de honeypots distribuidas entre varios usuarios.

2.4.3 Clasificación según su propósito

2.4.3.1 Honeypots de producción

Enfocados a distraer a los atacantes del acceso al sistema real alertan a los administradores de un ataque malicioso en la red para que ellos tomen medidas. Una vez generada la alerta se puede proceder a cambiar los Firewalls o las reglas de los IDS.
Por supuesto en este modelo no significa que el honeypot sea la vanguardia de la defensa del sistema, sería un cebo que actúa como sensor en caso de ataque.

2.4.3.2 Honeypots de investigación

Su objetivo es el de recopilar nuevas metodologías y tipos de ataques, intentando que sean atractivos para los hackers de forma que estos se mantengan dentro del sistema del honeypot. Resulta una herramienta muy eficaz para el estudio de la informática forense. Sin embargo, pocas entidades apoyan el uso de honeypots para la investigación debido a la creencia de que suponen un coste y no una inversión.

Este modelo de honeypot requiere adicionalmente que posea una mayor verosimilitud que los enfocados a producción, ya que el objetivo primordial es que el atacante muestre todas las técnicas posibles para la explotación de vulnerabilidades.
Nuestro proyecto pertenece a esta categoría debido a que el objetivo principal es el estudio de los ataques informáticos y no la defensa del sistema.

2.4.4 Clasificación según el hardware donde se despliegan

2.4.4.1 Honeypots físicos

Son una única máquina física corriendo un sistema operativo real con una única dirección IP. Están siempre orientados a ser honeypots de alta interacción y normalmente son menos prácticos que los virtuales debido a que están limitados a una única dirección IP y requieren un mayor mantenimiento físico.

2.4.4.2 Honeypots virtuales

Se implementan varios honeypots en una misma máquina física usando, por ejemplo, varias máquinas virtuales dentro de la máquina host. Esto permite emular varios honeypots que tengan distintas direcciones IP.

2.5 Ubicación del Honeypot

Otro de los aspectos que se debe tener en cuenta a la hora de desplegar Honeypots, es en qué lugar de la red de la organización se ubicará. Cada localización aporta una serie

ventajas y desventajas que hay que tener en cuenta para maximizar la efectividad y el número de ataques que reciba el honeypot.

Dependerá, sobre todo, el origen del objetivo del análisis pues los honeypots son capaces de recibir tanto ataques internos como externos, y la localización de este depende directamente de los tipos de ataque que recibirá[18].

2.5.1 Antes del Firewall

Desde esta posición el honeypot está en contacto directo con los atacantes externos, lo que permite registrar más ataques de los que serían filtrados por el firewall, formando la primera línea de defensa de nuestra red.

Este modelo permite obtener una estadística real sobre los ataques que se realizan a nuestra red desde otra red externa. Sin embargo, con este modelo no se pueden captar los ataques internos que pueda sufrir nuestra red.

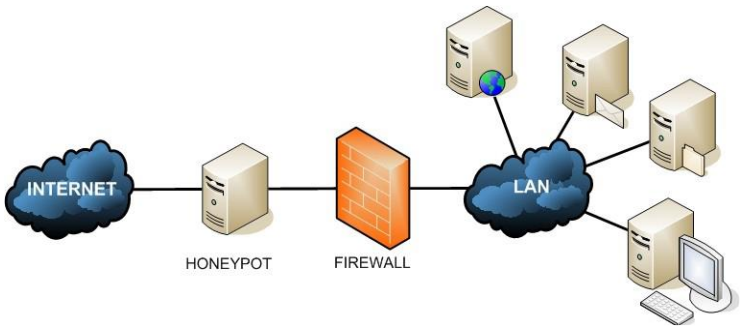


Figura 2-4: Honeypot delante del Firewall

2.5.2 Después del Firewall

Se sitúa al honeypot detrás del firewall en la misma red local que el resto de nuestro sistema lo que permite que el honeypot recopile ataques internos. A su vez, el hecho de que el Firewall filtre ataques impide que el honeypot recopile tanta información como lo haría fuera de este. Si enfocamos el honeypot a investigación estaremos perdiendo bastantes datos relevantes.

Sin embargo, esto puede servir para blindar aún más incluso el Firewall o detectar si está mal configurado pues si se registra un nuevo ataque en el honeypot es porque obviamente el Firewall lo ha admitido y se deberá crear una nueva regla de filtrado.

Inicialmente nuestro proyecto se desplegará con esta configuración, y en la fase de validación se pondrán parte de nuestros honeypots delante del Firewall.

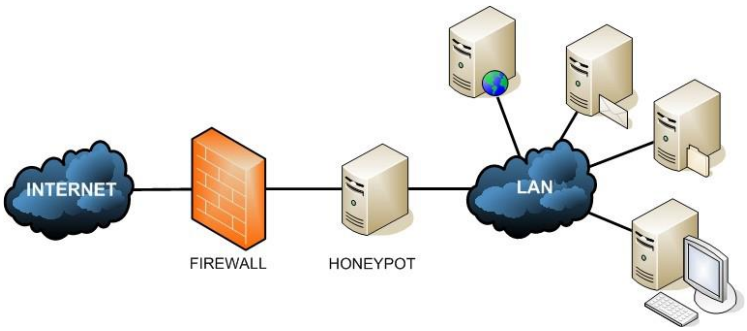


Figura 2-5: Honeypot detrás del Firewall

2.5.3 En una DMZ

Una zona desmilitarizada (conocida también como DMZ, sigla en inglés de *Demilitarized Zone*) o red perimetral es una zona segura que se ubica entre la red interna de una organización y una red externa, generalmente en Internet. El objetivo de una DMZ es que las conexiones desde la red interna y la externa a la DMZ estén permitidas, mientras que en general las conexiones desde la DMZ solo se permitan a la red externa, es decir, los equipos (hosts) en la DMZ no pueden conectar con la red interna.

Si se sitúa el honeypot en una DMZ se evita que si existe un IDS configurado en la red las alertas generadas por el tráfico del honeypot no afecten al IDS, pues el segmento de red donde situamos el honeypot (DMZ) está fuera del alcance del IDS.
La principal desventaja de este modelo es que se pierde cierto tráfico de ataques internos, pues para acceder al honeypot habría que hacerlo de forma explícita.

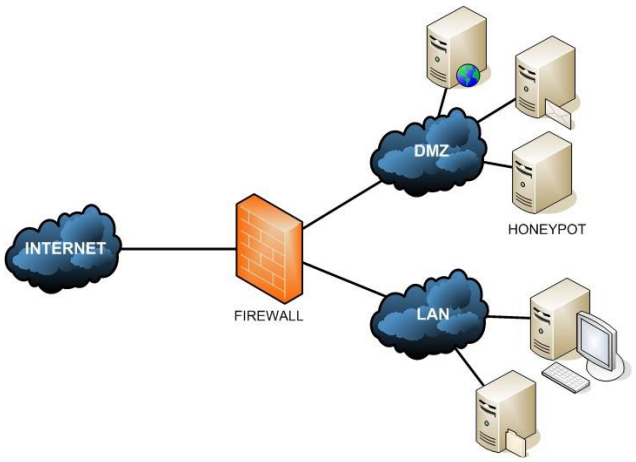


Figura 2-6: Honeypot en una DMZ

2.6 Análisis: Honeypots actuales

A continuación, se muestra una lista de los honeypots más relevantes en la actualidad y que hemos estudiado para la posible incorporación en la honeynet.

2.6.1 Kippo

Honeypot de media interacción que simula un servidor SSH [19]. Desarrollado en Python e inspirado en Kojoney [20], un honeypot de baja interacción que también simula un servidor SSH. Kippo resulta atractivo por ser fácil de configurar, y está enfocado a capturar ataques de fuerza bruta y todos los comandos que ejecute el atacante por Shell.

Kippo cuenta además con las siguientes características[21]:

- Simula un sistema de archivos en el que el atacante es capaz de agregar y borrar archivos. Un sistema completamente falso que se parece a una instalación de Debian 5.0.
- Tiene la posibilidad de añadir documentos al sistema simulado para que le resulte más real al atacante.
- Las sesiones que se registran durante los ataques pueden ser reproducidas posteriormente con los tiempos originales.
- Guarda todos los archivos que el atacante descarga desde el honeypot (con los comandos "wget" y "curl") para inspeccionarlos después.
- Simula la finalización de la sesión SSH: el honeypot realmente no finaliza la sesión y le muestra al atacante otra consola Shell distinta para así seguir monitorizando los movimientos del atacante.

2.6.2 Cowrie

Desarrollado por Michel Oosterhof [22], Cowrie proviene a partir de una rama del proyecto de Kippo que se ha hecho independiente y aún recibe actualizaciones.
Además de todas las funcionalidades que tiene Kippo, Cowrie cuenta con otras funcionalidades propias:

- Soporte de SFTP y SCP para la subida de archivos.
- Soporte de comandos de ejecución por SSH.
- Permite hacer login a través de TCP por un Proxy SSH.
- Permite guardar registros en formato JSON para una mayor facilidad de procesamiento.
- Acepta conexiones SMTP para dirigirlas a un honeypot de SMTP.
- Más comandos adicionales soportados.

2.6.3 Dionaea

Sucesor directo del honeypot Nepenthes (ya en desuso), Dionaea[23] es un honeypot de baja interacción que soporta múltiples protocolos.
Algunos de ellos son[24]:

- Server Message Block (SMB): es un protocolo de red que permite compartir archivos, impresoras, etcétera, entre nodos de una red de computadoras que usan el sistema operativo Microsoft Windows. Es el protocolo principal de Dionaea, un protocolo conocido por tener varios bugs explotables y objetivo de gusanos.

- Hypertext Transfer Protocol (HTTP): es el protocolo de comunicación que permite las transferencias de información en la World Wide Web. Dionaea soporta tanto HTTP como HTTPS en el puerto 80.
- File Transfer Protocol (FTP): es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP. Asignado al puerto 21, Dionaea permite la creación de directorios y la subida y descarga de archivos.
- Trivial File Transfer Protocol (TFTP): es un protocolo de transferencia muy simple semejante a una versión básica de FTP. Viene asignado al puerto 60.
- Microsoft SQL Server (MSSQL): asignado al puerto 1433, Dionaea implementa el protocolo TDS (Tabular Data Stream) usado en los servidores SQL de Microsoft permitiendo hacer login y decodificando las consultas que se ejecutan en la base de datos.
- Voice over IP (VoIP): el protocolo VoIP que usa Dionaea es SIP (Session Initial Protocol). Este módulo no se conecta a un servidor VoIP, simplemente espera por mensajes SIP entrantes que registra como incidentes o volcados de datos y responde acordemente.

La principal característica de Dionaea es el uso de la librería LibEmu[25], que simula un procesador x86 con detección de shellcode. Esto permite a Dionaea guardar todos los archivos que el atacante haya cargado en el honeypot y sean considerados maliciosos.

2.6.4 Conpot, Glastopf y SNARE/TANNER

Desarrollados por la fundación MushMush[26], SNARE/TANNER son los sucesores directos de Glastopf, un honeypot que emula una aplicación web. En cambio, Conpot es un honeypot de ICS (Industrial Control Systems) diseñado para recopilar información sobre los motivos y métodos que los competidores usan para encontrar los sistemas de control industriales.

2.6.4.1 Conpot

Conpot [27] soporta protocolos específicos de ICS como Modbus [28] muy populares para comunicaciones entre sistemas industriales y sus controladores. Cuando se implementaron estos protocolos únicamente estaban conectados a una red interna y no había necesidad de desarrollar seguridad para atacantes externos. Ahora que las compañías están conectadas a Internet, los sistemas que antes estaban aislados ahora aparecen en la red pública.

Conpot simula una infraestructura industrial completamente configurable para que el atacante piense que es real. Además, permite la integración de una HMI (*Human Machine Interface* [29]) para aumentar la visibilidad del honeypot.

2.6.4.2 Glastopf

Glastopf [30][31] es un honeypot de baja interacción que simula una aplicación web. Su principio de funcionamiento es responder al atacante con una respuesta que resulte realista y convincente, con el fin de provocar un ataque. Este Honeypot se centra en la emulación de los tipos de ataque en el lugar de las aplicaciones web en particular, que hace que sea versátil y fácil de mantener.

Algunas de las principales características de Glastopf son:

- Capacidad para guardar registros en SQLite o MySQL.
- Soporte para HPFeeds[32] (implementa un protocolo publicador-suscriptor).
- Permite inyección SQL.
- Extrae los dorks de las peticiones para poder indexarse mejor, lo que le otorga mayor visibilidad según es atacado.
- Contiene varias vulnerabilidades conocidas de los servicios web.

2.6.4.3 SNARE/TANNER

Sucesores de Glastopf, SNARE [33] se encarga de sensorizar las actividades maliciosas en una aplicación web. TANNER [34] se encarga de evaluar las actividades que recibe SNARE y decide cómo SNARE debe responder al cliente. Esto permite que se puedan desplegar varios sensores SNARE y un único TANNER en lugar de parejas de SNARE/TANNER que es lo equivalente a una instancia de Glastopf.

2.6.5 Twisted-Honeypots

Desarrollado por Lanjelot, Twisted-Honeypots[35] es un honeypot de baja interacción basado en el motor Twisted [36].
Soporta los protocolos SSH, FTP y Telnet, pero únicamente recoge passwords y no ofrece una consola con comandos al atacante. Debido a las carencias para la investigación que tiene Twisted-Honeypots, es recomendable enfocarlo para la disuasión de los atacantes o para conocer las passwords más utilizadas.

3 Análisis y Diseño

3.1 Etapas del proyecto

Para la creación de la honeynet se han realizado las siguientes etapas:

I. Estudio de los honeypots actuales para la incorporación a la honeynet:

Análisis de los honeypots disponibles para comprobar la compatibilidad que pueden tener usando una misma dirección IP y posteriormente instalarlos en máquinas virtuales para que empiecen a recibir ataques. Estuvieron encendidos 24h para simular un servicio web real.

Para la creación de la honeynet local se usaron los siguientes recursos:

- Máquinas virtuales gestionadas por VirtualBox.
- Dos tipos de distribuciones Linux (Debían y Ubuntu).
- 4 direcciones IP estáticas cedidas por la UAM.
- Sistema Operativo Ubuntu para el host.
- Base de datos relacional PostgreSQL.

II. Diseño y desarrollo de la infraestructura:

Diseño de las distintas configuraciones del sistema para realizar los análisis y posterior desarrollo de estas. Las configuraciones que se implementaron fueron para cumplir tres propósitos:

- Análisis de los honeypots: se desarrolló una configuración para el estudio de los distintos honeypots.
- Análisis del Firewall: se configuró el sistema para comprobar la efectividad del Firewall externo de la UAM.
- Creación de una red global: se diseñó una infraestructura que permita la inclusión de nuevos sensores a la red distribuidos en distintas ubicaciones.

III. Pentesting (O simulación de intrusión) en los honeypots para comprobar sus capacidades:

Se procedió a probar con herramientas enfocadas a auditorías de seguridad cómo generan los registros los distintos honeypots. Como podemos comprobar el sistema que estamos simulando en cualquier momento, el tipo de auditoría se adecuó al de caja negra. Las herramientas que se utilizaron para cada nivel son las siguientes:

- Nivel Footprinting: La capa más superficial. Se busca el dominio de cada máquina virtual para ver la información que puede obtener un usuario común. Se usan herramientas como Anubis, Google Dorks, Netcraft, Whois e IPtools.
- Nivel Fingerprinting: un poco más profunda que la capa de Footprinting. Se usan herramientas un poco más especializadas como Maltego, FOCA o Nmap.

- Nivel de análisis de vulnerabilidades: se usan herramientas automáticas (Nessus, Netsparker o Accunetix) para encontrar las vulnerabilidades comunes que suelen tener los servidores web.
- Nivel de ganancia acceso y escalamiento de privilegios: no nos centramos mucho en este nivel, pero sí que es interesante ver cómo otros usuarios intentarían llegar a tomar control sobre nuestro sistema o si intentarían meter algún tipo malware. Se probó a meter algún malware para ver cómo funciona.

IV. Securización y ocultación de los honeypots:

Una vez analizados los honeypots se procedió a ocultarlos a los atacantes para que no sospechen y crean que están atacando un sistema real.

Algunas medidas que se tomaron:

- Renombramiento de directorios.
- Habilitación de comandos específicos.
- Cambio de aspecto (del que viene por defecto) en aquellos que sean web.

V. Visualización de las capturas recogidas:

Para comprobar los resultados obtenidos y analizarlos de una forma cómoda que permita la monitorización completa de los honeypots, se implementó un servicio web para la consulta los datos. Este servicio web posteriormente estará disponible para los usuarios que participen en nuestra red.

VI. Análisis de los ataques:

Realizadas las mejoras en los honeypots, se analizaron los ataques que reciben los honeypots con el fin de normalizarlos y almacenarlos en una base de datos PostgreSQL.

En este caso se generan estadísticas y se aplicará informática forense para analizar los tipos de ataques y malware.

VII. Validación del sistema a través del Firewall de la UAM:

Se analizó el firewall de la UAM poniendo los honeypots en espejo (los mismos honeypots con la misma configuración dentro y fuera del firewall). De esta forma se comprueba la efectividad del Firewall y además se puede validar el sistema desplegado.

VIII. Desarrollo de una máquina virtual distribuible para la creación de nuevos sensores:

La fase final consta de la creación de un archivo ejecutable que instala una máquina virtual con todos los honeypots en el equipo del usuario. Se registran los datos tanto localmente en la máquina del usuario como remotamente en nuestra base de datos centralizada.

Se muestran como opciones de instalación los distintos honeypots que hemos configurado para nuestra Honeynet de forma simple y lo más transparente posible para el usuario.

3.2 Diseño de la infraestructura

A continuación, se muestra los diseños realizados según el propósito de su utilización.

3.2.1 Diseño para el análisis de los honeypots

Este diseño permite la instalación de varios honeypots en una única máquina real para su posterior estudio. Con este diseño se realizaron las pruebas de pentesting que ayudaron a la correcta configuración de los honeypots.

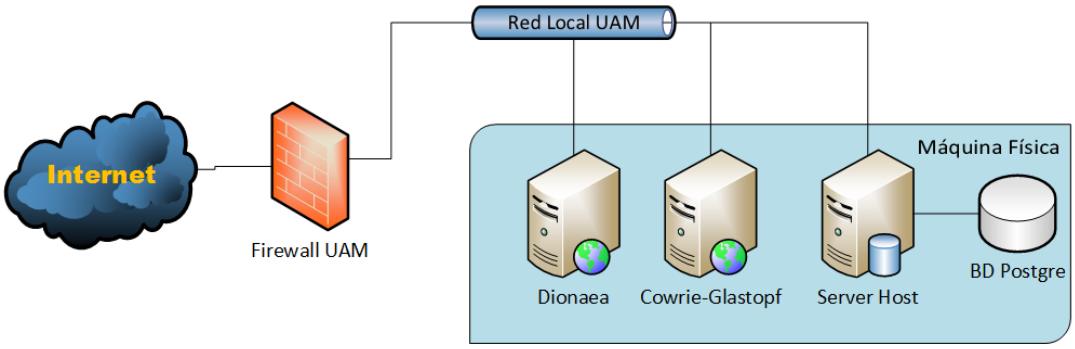


Figura 3-1: Diseño para el análisis de los honeypots

Como se ve en la Figura 3-1 se han asignado dos direcciones IP a dos máquinas virtuales que son las que contienen los distintos honeypots. Todas estas máquinas virtuales se encuentran detrás del Firewall de la UAM. A la hora de desplegar los honeypot, hay que tener en cuenta que pueden tener colisiones por usar el mismo servicio y/o puerto. Por ejemplo: Glastopf no pude ir en la misma máquina virtual que Dionaea pues ambos ofrecen el servicio HTTP que normalmente va asignado al puerto 80.

La distribución que se ha usado es la siguiente:

- Cowrie-Glastopf: Glastopf simula un servicio HTTP y Cowrie de FTP y SSH, al no tener colisiones en sus servicios se pueden situar en una misma máquina virtual.
- Dionaea: como Dionaea incluye los servicios FTP y HTTP además de otros muchos protocolos se ha decidido colocarlo solo en una Máquina Virtual.

Los datos recogidos por los honeypots se envían a una base de datos alojada en el host de las máquinas virtuales para su posterior análisis.

Se han seleccionado estos honeypots debido a que simulan servicios conocidos por los atacantes y no tan específicos como protocolos para una infraestructura SCADA como simula Conpot.

También se ha elegido a Glastopf antes que SNARE/TANNER debido a que simulan lo mismo y la versión de Glastopf que existe es más estable que la de SNARE/TANNER.

3.2.2 Diseño para el análisis del Firewall de la UAM

El objetivo de esta infraestructura es analizar el Firewall de la UAM. Para ello se han seleccionado dos de las máquinas virtuales usadas en el apartado anterior y se ha generado una copia idéntica de cada una.

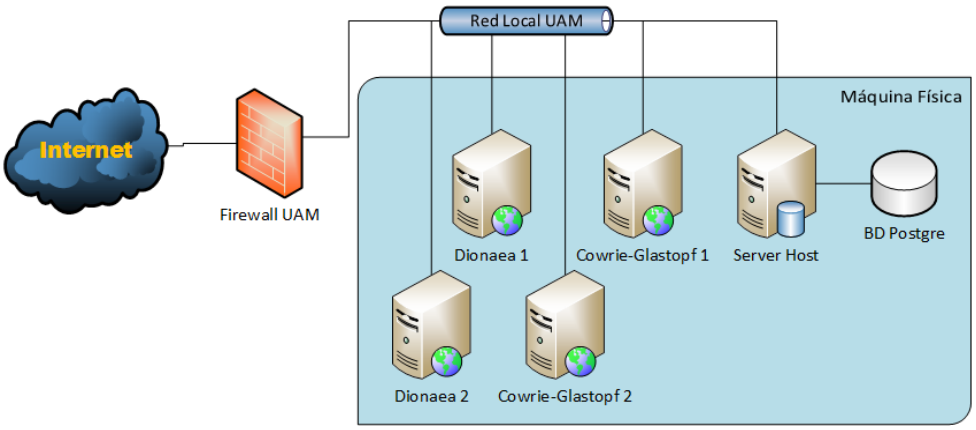


Figura 3-2: Diseño análisis Firewall inicial

Como se puede ver en la Figura 3-2, las máquinas virtuales seleccionadas han sido las que contienen a Dionaea y Cowrie-Glastopf. Se han escogido estas dos principalmente porque son las que contienen los honeypots con más servicios y los que usualmente son más atacados.

Este análisis se ha dividido en dos fases:

- Fase de control: se colocan todos los honeypots detrás del Firewall y se comprueba que reciben un número de ataques similares. La distribución usada corresponde con la Figura 3-2.
- Fase experimental: se sitúan los honeypots originales detrás del Firewall y las copias delante de este. De esta forma se comprueban los ataques que previene el Firewall comparando los resultados obtenidos por los honeypots. La distribución de la red para esta fase corresponde con la Figura 3-3.

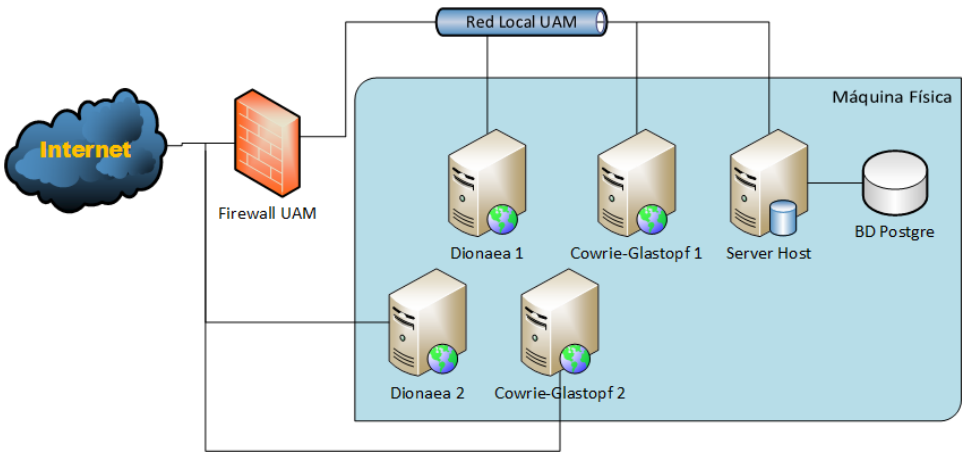


Figura 3-3: Diseño análisis Firewall final

3.2.3 Diseño para la creación de una red global

Este último diseño tiene como objetivo describir el funcionamiento de la red global de honeypots. Como se observa en la Figura 3-4, se mantiene el diseño local de la etapa previa del análisis de los honeypots además de una red externa formada por usuarios que ejecutan una máquina virtual con honeypots cuyas capturas se registran en la base de datos que se encuentra en la UAM.

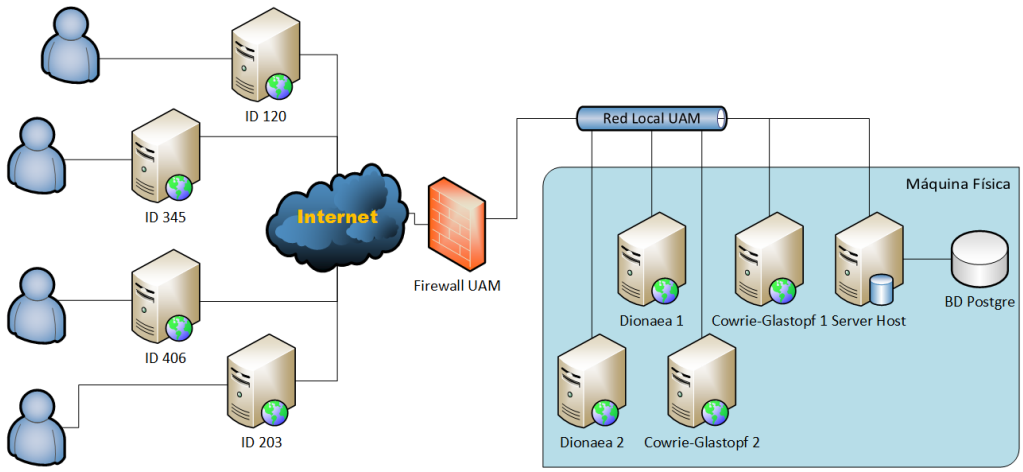


Figura 3-4: Diagrama de red global

Cada una de las máquinas virtuales que sea ejecutada por los usuarios contiene un identificador único, lo que permite identificar qué usuarios consiguen más capturas para posteriormente adoptar la configuración que hayan usado dichos usuarios en sus máquinas virtuales.

3.3 Pentesting (o simulación de intrusión) en los honeypots

Pentesting (del inglés *Penetration Testing*) [37, p. 8], es la metodología, proceso y procedimientos usados por los *testers* para evadir las protecciones de los sistemas de información. Este tipo de testeo está asociado con la evaluación de las configuraciones técnicas, administrativas y operacionales y los controles del sistema. Normalmente los test de penetración sólo evalúan la seguridad del sistema de información tal como está construido.

En este apartado se probó con herramientas típicamente usadas por hackers la robustez de los honeypots, su realismo y posteriormente cómo los honeypots registran dichos ataques.

Para ello se pasó por distintos niveles de seguridad y se observó qué tipo de información aportan nos aportan los honeypots.

3.3.1 Nivel Footprinting

También conocida como la fase de reconocimiento. Esta fase se centra en conocer toda la información que es pública acerca del sistema que se quiere atacar: búsqueda en Internet, escaneos pasivos… En esta fase, el *tester* sólo se centra en documentar tanta información del objetivo como sea posible.

Para este nivel se utilizaron métodos como el acceso directo al servicio, la web de búsquedas Shodan [38] y la herramienta FOCA [39].

Shodan ha añadido recientemente una nueva herramienta llamada *"Honeypot Or Not?"* [40] que analiza directamente la dirección IP para comprobar si lo reconoce como honeypot o no. Esta funcionalidad, se usó también en la sección de pruebas.

3.3.2 Nivel Fingerprinting

Fase de escaneo. En esta fase el objetivo es obtener una información más detallada sobre el sistema que se está analizando. A partir de la información obtenida en la fase de reconocimiento y con la ayuda de herramientas especializadas, se consigue la información necesaria para encontrar las vulnerabilidades que nos den acceso al sistema en la siguiente fase.

Con estos dos niveles se consigue analizar la percepción que un atacante tiene sobre el honeypot. En los siguientes niveles se comprobará la capacidad de nuestro honeypot para capturar ataques.

Para este nivel se le da un gran peso a la herramienta Nmap [41], una aplicación de código abierto que escanea los puertos de una dirección IP además de otras utilidades. Se ha decido usar esta aplicación debido a su uso extendido entre los profesionales del sector de la seguridad informática.

3.3.3 Nivel de análisis de vulnerabilidades

En este nivel se buscan las vulnerabilidades que nos pueden dar acceso al sistema y se puedan explotar o informar de ellas. Es el paso anterior de obtener control sobre el sistema objetivo.

3.3.4 Nivel de ganancia acceso y escalamiento de privilegios

El objetivo de este nivel es obtener el mayor número de privilegios posibles. Además, normalmente los hackers dejan alguna puerta abierta para acceder en un futuro de una forma más sencilla.

3.3.5 Nivel Final

Dependiendo de la ética, una vez analizado el sistema y explotado sus vulnerabilidades se pueden reportar al administrador o vender dicha información. En este proyecto no se llegará a esta fase pues es innecesaria.

3.4 Securización y ocultación de los honeypots

Una vez realizadas las pruebas de pentesting de los niveles de Footprinting y Fingerprinting se modificarán las configuraciones de los honeypots para que fueran más realistas y no sean detectados por los atacantes como honeypots.

3.5 Visualización de los datos

Para acceder a la información que recogen los honeypots se utilizó la tecnología ELK Stack (o Elastic Stack) [42], que permite obtener una visualización de los datos más interactiva y cómoda. Su interfaz web es bastante intuitiva, permitiendo que los nuevos usuarios que participen en nuestra red no tengan mucha dificultad para visualizar los datos como ellos desean.

4 Desarrollo

4.1 Desarrollo de la infraestructura

En este apartado se mostrarán el software y configuraciones usadas durante el proyecto.

4.1.1 Software base

Para el desarrollo de la infraestructura se ha decidido usar la siguiente configuración:

- **Base de datos:** se ha decidido usar una base de datos relacional en formato PostgreSQL [43], debido a que es software libre, no necesita licencia y permite la creación de bases de datos enormes con una alta eficiencia.
- **Máquinas virtuales:** para desplegar las máquinas virtuales se ha decidido usar VirtualBox [44], permite la ejecución de varias máquinas virtuales simultáneamente, la asignación de recursos a cada máquina virtual y una interacción cómoda entre el host y las máquinas virtuales.
- **Sistemas operativos**: los sistemas operativos usados en las máquinas virtuales han sido Debian 7.10 [45] y Ubuntu 14.04 [46]. Debido a que Dionaea sólo tiene soporte oficial para la versión 14.04 de Ubuntu, se ha decidido usar esta versión y no una superior. Debian se ha seleccionado para comprobar si los honeypots funcionaban en otras distribuciones Linux además de Ubuntu.
- **Direcciones IP**: las direcciones IP usadas son estáticas. Además, cada una tiene asignada un dominio. Las llamaremos IP1, IP2, IP3 e IP4 para referirnos a ellas en los siguientes apartados.

4.1.2 Desarrollo del diseño para el análisis de los honeypots

Para llevar a cabo el despliegue de la red para analizar los honeypots se han seguido los siguientes pasos:

- **Instalación de los honeypots**: se han instalado los honeypots como se muestra en la Figura 3-1. Para más detalles sobre cómo se realizó su instalación consultar los anexos A.1 para la instalación de Dionaea, y A.3 y A.4 para los de Cowrie y Glastopf. También se llegó a instalar Kippo (anexo A.2), pero resultó ser muy similar a Cowrie y con menos características, por ello se descartó.
- **Asignación de direcciones IP**: posteriormente se asignaron dos direcciones IP estáticas, una para cada máquina virtual. A la máquina virtual donde está alojada Dionaea se le asignó la IP1 y a la otra máquina que contiene a Glastopf y Cowrie la IP2.

Una vez instalados y puestos en ejecución, se analizaron los honeypots y se pasaron las pruebas de pentesting para posteriormente ocultarlos a los atacantes como si fuesen sistemas reales.

Además, los ataques recibidos durante el tiempo que estuvieron activos sirvieron para la fase de validación de nuestro sistema.

4.2 Desarrollo del diseño para el análisis del Firewall de la UAM

Para esta fase se han utilizado los honeypots que se desplegaron en la fase de análisis de los honeypots una vez que pasaron las pruebas de pentesting y fueron ocultados.
En esta sección se duplicaron dichos honeypots clonando las máquinas virtuales y utilizando otras dos nuevas direcciones IP.

Para este diseño se renombró a los honeypots originales que contenían las direcciones IP1 e IP2 a Dionaea 1 y Glastopf-Cowrie 1. A las máquinas clonadas se les asignó los nombres de Dionaea 2, con la dirección IP3 y Glastopf-Cowrie 2, con la dirección IP4. Se puede ver una representación del sistema en la Figura 4-1:

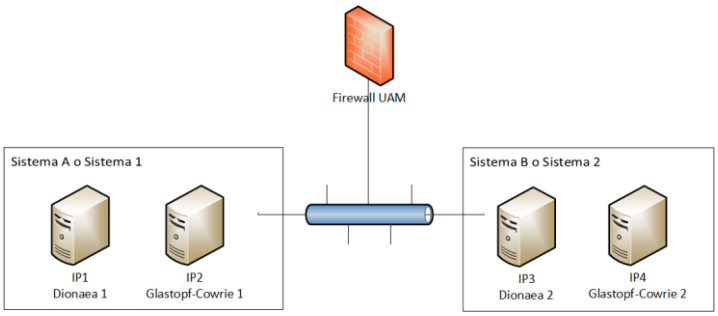


Figura 4-1: Diagrama Desarrollo análisis Firewall UAM 1

Se dejó esta configuración durante los primeros 5 días de junio para comprobar el funcionamiento de las nuevas máquinas virtuales.
Una vez analizados los datos (apartado 5.1.2 de esta memoria) se decidió posicionar el sistema B fuera de la protección del Firewall, y así comprobar su efectividad. La Figura 4-2, representa la distribución del sistema fuera del Firewall:

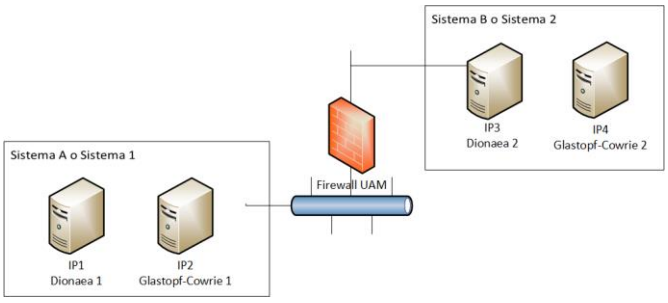


Figura 4-2: Diagrama Desarrollo análisis Firewall UAM 2

4.3 Pentesting y ocultación de los honeypots

En este apartado se realizaron distintas pruebas generalmente usadas para los test de penetración en sistemas reales, que ayudaron a detectar cómo de real es nuestro sistema. Se mostrarán los cambios realizados en los honeypots durante las pruebas para que posteriormente no fuesen detectados por estas.

Algunas de las herramientas utilizadas fueron:

- **Shodan**: Es un motor de búsqueda que muestra información sobre los servidores o routers que se están ejecutando en una dirección IP. Nos aporta información pública como los servicios que se están ejecutando en un servidor, su proveedor de internet o el dominio que tiene asignado.
- **Honeypot Or Not?**: Es una herramienta creada por el equipo de Shodan, simplemente se introduce una dirección IP y determina si es un sistema real o no. Pasar este test aporta mayor realismo a nuestro honeypot. Al igual que Shodan fue utilizada durante la fase de Footprinting.
- **FOCA**: una herramienta que tiene como función principal obtener todos los metadatos posibles de un dominio y analizarlos. De esta forma podemos descargar todos los documentos públicos de un dominio y ver qué usuarios los han creado, qué impresoras han utilizados y mucha más información. Pertenece a la fase de Fingerprinting.
- **Nmap**: herramienta de código abierto para el escaneo de puertos. Se ha escogido Nmap por ser muy popular entre los profesionales de la seguridad informática. Al igual que FOCA, Nmap pertenece a la fase de Fingerprinting.

Para el nivel de análisis de vulnerabilidades normalmente se usan herramientas automáticas para encontrar los fallos de configuración. En nuestro caso se ha enfocado a cómo un atacante intentaría aprovecharse de dichos fallos para acceder al sistema.
Durante este análisis se fueron modificando los honeypots para que no fuesen detectados por las herramientas ya mencionadas. Para ver el desarrollo completo de esta fase ver el anexo D.

4.4 Visualización de los datos y ELK Stack

En este apartado se mostrará el servicio web implementado para la visualización de los datos.

En primera instancia se probó a visualizar los datos con DionaeaFR [47], un servidor Front-End basado en Django [48]. Después de realizar la instalación en las máquinas virtuales como se muestra en el anexo A.5, se intentó reusar en la visualización de los datos de otros honeypots, con escaso éxito pues obligaba a pasarlos antes a una base de datos SQLite y no aportaba ningún tipo de seguridad. Se decidió entonces pasar a usar ELK Stack.

La arquitectura ELK Stack [42], cuya función principal es la monitorización de *logs* o registros, está compuesta por varios módulos:

- **Logstash**: procesa los *logs* que llegan al servidor y aplica filtros según el tipo de *log*.
- **Elasticsearch**: almacena en una base de datos los *logs* que ha procesado Logstash.
- **Kibana**: es una interfaz web que representa los *logs* almacenados en Elasticsearch de forma gráfica.
- **Filebeat**: se instala en los sensores y envía los *logs* que se generan al servidor donde se encuentre Elasticsearch.

A continuación, la Figura 4-3 muestra el funcionamiento global de la arquitectura:

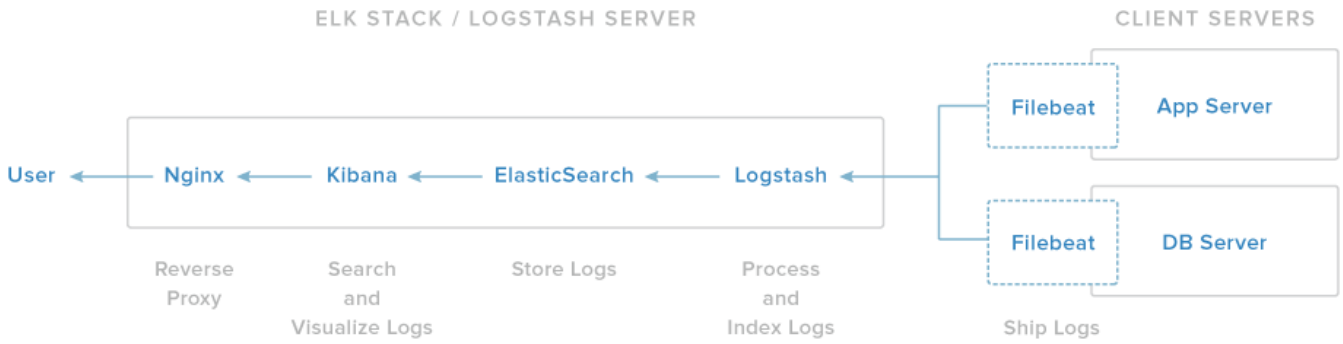


Figura 4-3: Arquitectura ELK Stack

En nuestra red se ha instalado Filebeat en cada máquina virtual asignando un identificador a cada máquina en el fichero de configuración. En el servidor central podemos saber desde qué máquina provienen los logs con dicho identificador y de qué tipo son. Filebeat sólo reenvía *logs* en archivos de texto y no dentro de bases de datos, por ello para Dionaea y Glastopf se crearon unos scripts en Python que generan archivos de texto a partir de las tuplas que se encuentran en sus respectivas bases de datos. Estos se scripts se ejecutan automáticamente usando crontab [49] y puede verse su configuración en el anexo B.3.

Logstash, Elasticsearch y Kibana se instalaron en el host, siendo este nuestro servidor central.

Logstash recibe los *logs* de Filebeat y ordena los datos por campos indicando qué tipo de datos contienen y su formato. Posteriormente envía los datos filtrados a Elasticsearch para que los almacene en su base de datos

Elasticsearch no es exactamente una base de datos SQL [50], pero si tiene una gran similitud y puede exportar los datos en un archivo CSV [51], permitiendo que se agreguen a una base de datos SQL.

Finalmente, a través de Kibana podemos acceder a los datos almacenados en Elasticsearch vía web a través de un usuario y contraseña. Los detalles de su instalación y configuración se muestran en los anexos A.7 y B.2.

4.5 Creación de una base de datos SQL

Una vez desplegado nuestro sistema se almacenarán los ataques en una base de datos SQL, concretamente en PostgreSQL. Todos los ataques se almacenan en una sola tabla y dependiendo del tipo de ataque y del honeypot se guardan los datos adicionales en otras tablas.

La primera opción fue pasar directamente los datos almacenados en las bases de datos de Dionaea y Glastopf, pero esto obligaba a realizar una conexión activa al honeypot desde el servidor para consultarlas. Habiendo instalado ELK Stack, podemos pasar por un fichero CSV todos los datos almacenados en Elasticsearch y agregarlos a la base de datos PostgreSQL. De esta forma podemos eliminar los datos más viejos de Elasticsearch y hacer que vaya más fluido, mientras mantenemos dichos datos en nuestra base de datos SQL.

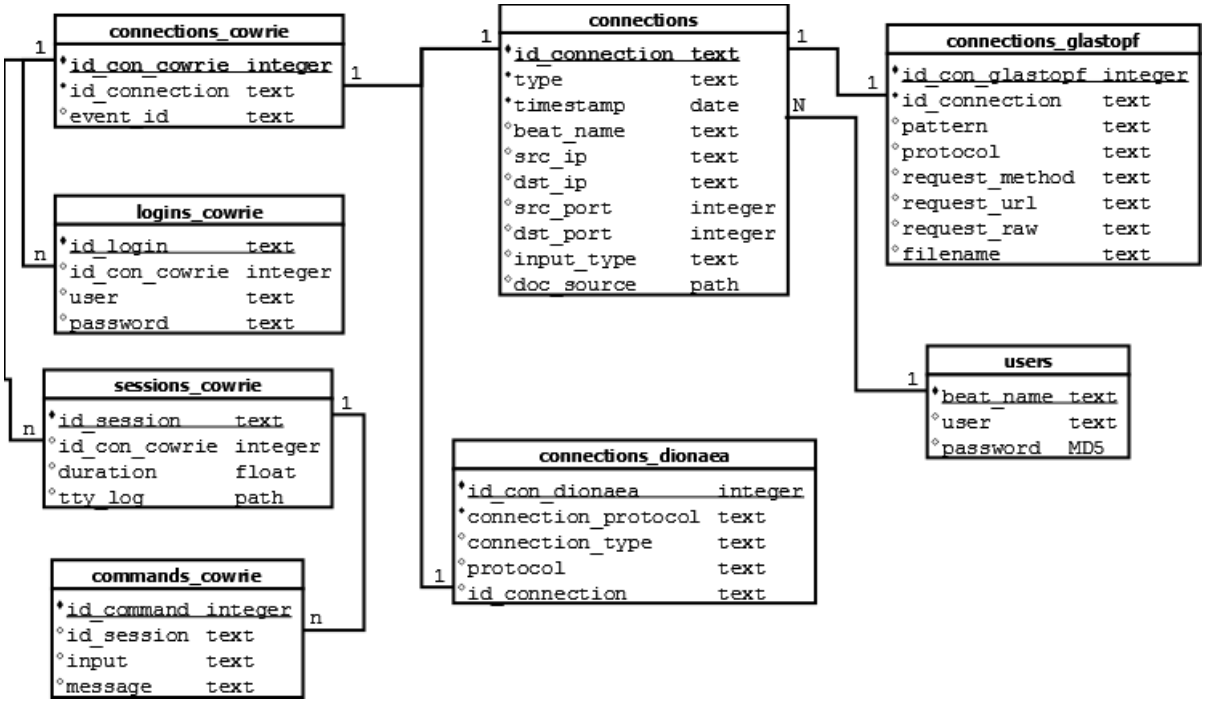


Figura 4-4: Diagrama Base de Datos SQL

Como se puede ver en la Figura 4-4, la base de datos está compuesta por una tabla central que recoge todas las conexiones ("connections") y dependiendo del tipo de honeypot existen sub-tablas que aportan mayor información a esa conexión.

Cowrie por ejemplo muestra el tipo de conexión que es:

- Si es un intento de login se consultaría la tabla "logins_cowrie" para ver el usuario y contraseña que ha usado.
- En caso de que se haya realizado la conexión con éxito la tabla "sessions_cowrie" muestra la duración de esta y el archivo tty de dicha conexión. Además, la tabla "commands_cowrie" nos muestra los comandos que realizó el atacante y cuál fue la respuesta del honeypot.

La tabla de "connections_dionaea" hace referencia a los datos que ofrece Dionaea al recibir conexiones:

- El campo "connection_protocol" muestra el protocolo de la capa de aplicación que se ha usado durante la conexión (HTTP, MSSQL, SIP …).
- Si la conexión ha sido rechazada o aceptada se puede ver en el campo "connection_type".
- El campo "protocol" muestra el tipo de protocolo de transporte usado en la conexión en (TCP, UDP, TLS…).

Glastopf también tiene su propia tabla, “connections_glastopf”, donde podemos consultar el patrón de la conexión (Login, SQL Injection, HEAD…), el protocolo (TCP, UDP) y los datos de la url que el atacante ha utilizado.

Por último, la tabla “users” hace referencia a los usuarios que pueden conectarse a la base de datos y tienen un identificador único (“beat_name”) con el que registran ataques desde su propia red y reenvían a nuestra base de datos Elasticsearch.
Como se puede observar la base de datos está en 3FN pues no hay dependencias funcionales transitivas con atributos que no son claves primarias.

4.6 Desarrollo de una máquina virtual distribuible para la creación de nuevos sensores

Finalmente, se desarrolló una máquina virtual distribuible que permite la incorporación de nuevos usuarios a nuestra red de forma automática. Para ello se replicó las instalaciones y configuraciones de los honeypots utilizados en nuestra red en una única máquina virtual que se entrega a los nuevos usuarios.

Se ha escogido el uso de máquinas virtuales en lugar de contenedores o *dockers*[52], debido a que la virtualización que tienen las máquinas virtuales crea un mayor aislamiento con el host que los *dockers*. Otro punto a favor, es que los *dockers* están enfocados a ejecutar varios a la vez en un mismo host, y en nuestro caso únicamente necesitamos ejecutar una máquina virtual con los tres honeypots.

La ventaja de tener ejecutando a Logstash en nuestro servidor es que escucha todo lo que le envía cualquier instancia de Filebeat a través del puerto que tiene asignado. En caso de que los datos recibidos no coincidan con los especificados en el filtro, se descartan. También se puede bloquear una conexión con el servidor cuando una IP o usuario no registrado en nuestra base de datos intente acceder por ese puerto.

La configuración base es asignar el usuario “Guest” a una máquina virtual hermética en la que el usuario no pueda realizar cambios y de esta forma nosotros controlar quién se conecta. Los detalles de la instalación se pueden ver en el anexo E. En un futuro, cuando el sistema de usuario esté pulido, se puede contemplar la posibilidad de hacer la máquina virtual distribuible más flexible para el usuario.

Inicialmente la máquina virtual se entregará a usuarios de confianza a través de un enlace de descarga. Posteriormente se puede aumentar la visibilidad del proyecto creando una página web donde los usuarios puedan descargar la máquina virtual y consultar el estado de su cuenta.

5 Integración, pruebas y resultados

5.1 Análisis del Firewall de la UAM

Para esta fase primero se han analizado las capturas en la etapa de control para comprobar si las capturas en los dos honeypots son similares y en la segunda etapa, las diferencias entre las capturas realizadas por los honeypots que se encuentran delante del Firewall y los que se encuentran detrás.

Se mostrará además un análisis previo de las capturas recogidas por los honeypots antes de ser divididos en parejas.
Dado el gran volumen de gráficos y tablas generadas, se han recogido todos en el anexo C.

5.1.1 Análisis capturas Etapa previa al experimento

Esta etapa abarca todo el tiempo que estuvieron funcionando los honeypots antes de ser divididos en parejas para el análisis de Firewall, por ello sólo aparece un honeypot de cada clase.

5.1.1.1 Análisis estadístico

Los honeypots estuvieron en funcionamiento desde mediados de noviembre de 2016 y debido a funciones de mantenimiento del servidor, Cowrie y Glastopf estuvieron parados durante los meses de marzo y abril como se puede observar en las secciones C.1.Gráficos Resultados 1 y Dionaea durante el mes de marzo.

Durante el tiempo en que estuvieron operativos el honeypot Cowrie llegó a recoger 7.881 conexiones, Glastopf 18.725.018 y Dionaea 45.864. Como se puede ver en la sección C.1.Gráficos Resultados 2, la mayoría de las conexiones provienen de Estados Unidos y China a excepción de Glastopf que ha recibido una gran cantidad de conexiones desde las Islas Seychelles. Los ataques provenientes de Estados Unidos y China son algo previsible pues son dos de los países que más ataques realizan durante todos los años. Sin embargo, los ataques desde las Seychelles es algo excepcional y se estudiará en el análisis forense.

Si nos fijamos en los resultados mostrados en las secciones: C.1.Gráficos Resultados 3, 4, y 5 se pueden observar las direcciones IP que más se han conectado a nuestros honeypots. Resalta la conexión con IP “93.174.93.103” pues compone el 78% del tráfico dirigido a Glastopf con más de 14 millones de conexiones, además de ser la única dirección proveniente de las Seychelles. También se analizarán posteriormente en el análisis forense las direcciones IP que comienzan por “104.192.3.*” pues también componen un gran porcentaje de las conexiones a Glastopf.

En cuanto a Cowrie y Dionaea ambos tienen una tasa de conexiones más distribuida, aunque también poseen una dirección que resalta por encima de las demás como es la “202.92.17.140” en el caso de Dionaea o la “217.23.10.181” en el caso de Cowrie.

En la tabla - Top Combinaciones Usuario-Contraseña (E. Previa), podemos ver las combinaciones usuario-contraseña más utilizadas por los atacantes. La mayoría son

bastante conocidas por todos como los usuarios “root” o “admin” o las contraseñas “12345” y “password”. Esto es debido a que aún existen administradores que utilizan este tipo de contraseñas y por ello los atacantes se dedican a buscar con bots sistemas que tengan estas passwords, pues es un ataque por fuerza bruta muy simple y que si acierta garantiza acceso al sistema víctima.

En los C.1.Gráficos Resultados 7 se puede ver que la mayoría de conexiones que se han realizado a Glastopf eran del tipo HEAD, este método HTTP es parecido a GET pero sólo se envía la cabecera del archivo. Si nos fijamos en la imagen de C.1.Gráficos Resultados 8 podemos ver algunos ejemplos de estas peticiones.

5.1.1.2 Análisis forense

Glastopf: primero vamos a analizar la dirección IP “93.174.93.103” que fue la que más conexiones realizó a Glastopf. Si nos fijamos en los patrones de las peticiones que realizó se observa que todas ellas eran con el método HEAD, un método que en principio no es ningún intento de penetración al servidor como puede ser un intento de login o de SQLInjection.
Si analizamos la dirección en Shodan podemos ver que está asignada al dominio “no-reverse-dns-configured.com” y que la organización a la que está asignada “Novogara LTD” coincide con el IPS, así que podemos intuir que se trata de un router personal y no de una compañía profesional.

Otra información que nos proporciona Shodan son los servicios que ejecuta el servidor. En este caso llama la atención que tenga asignado el puerto 111 a un portmapper, cuyo objetivo es el escaneo de puertos de servidores. Por último, podemos corroborar que el objetivo de dicho servidor es espiar nuestros puertos viendo que en Cymon[53] fue reportado y puesto en listas negras por dicha actividad.

Ahora pasemos a las direcciones IP que empiezan por “104.192.3.*”. Si las buscamos en Shodan vemos que no aportan mucha información a parte de decir que tienen un servidor Apache y en Cymon pone que han sido reportadas y metidas en listas negras, pero sin poner el motivo.

Para salir de dudas y determinar el objetivo de dichas direcciones IP las buscamos en AbuseIPDB[54], podemos ver que han sido reportadas tanto por escaneo de puertos como por ataques DDoS.

Cowrie: en este caso analizaremos la dirección IP (“217.23.10.181”) con más conexiones a Cowrie.
Cowrie sólo ofrece servicio SSH y sólo registra ataques en el puerto asignado, si buscamos la dirección IP en Cymon y AbuseIPDB se observa que dicha dirección ya ha sido reportada por ataques de fuerza bruta por SSH. Shodan por otra parte nos indica que tiene otros servicios abiertos como un servidor web de Microsoft o un servicio para escritorio remoto.
Analizando los comandos ejecutados ese día no aparece que la dirección analizada intente ejecutar o descargar ningún archivo puede que se deba a que una vez conseguido el acceso intente ejecutar comandos desde otra dirección.

Dionaea: pasamos a analizar la dirección "202.92.17.140" que fue la que más conexiones realizó a Dionaea. Si la buscamos en Shodan o en Cymon vemos que no hay resultados para dicha IP. Sin embargo, en AbuseIPDB varios usuarios la han reportado como PortScaner o que ha hecho uso de PortMap. Podemos verificar que este tipo de ataque es el que ha efectuado en el honeypot Dionaea seleccionando dicha IP en Kibana para mostrar todas sus conexiones.

Se muestra que de las 863 conexiones sólo una está dirigida al servicio ftp y una al de http, el resto de conexiones son a puertos que no tienen asignados un servicio, por tanto, podemos concluir que ha realizado un escaneo en el honeypot.

5.1.2 Análisis capturas - ambos sistemas detrás del Firewall

En este apartado se muestran los ataques recibidos por los honeypots Cowrie, Glastopf y Dionaea y sus pares dentro del Firewall de la UAM.

Las capturas obtenidas son de los 5 primeros días de junio, cuando se inicializaron a la vez los pares de máquinas virtuales.

5.1.2.1 Análisis estadístico Cowrie 1 y 2

.
Como se puede ver en C.2.1.Gráficos Resultados 9, el número de ataques recibidos por las dos direcciones IP son similares, llegando a un total de 220 conexiones en la primera máquina y 206 en la segunda.

Las combinaciones de usuario-contraseña más populares que utilizan los atacantes son con el usuario: "admin" o "root" y la password: "1234" o "admin1234", se puede ver una lista más detallada en los C.2.1.Gráficos Resultados 10.

A continuación, se muestran en los C.2.1.Gráficos Resultados 12 y C.2.1.Gráficos Resultados 13 las direcciones IP que se han conectado a nuestros honeypots. Se puede observar que hay algunas direcciones IP similares como la "195.22.127.83" que ha accedido a las 2 máquinas.
Si buscamos dicha IP en Shodan nos encontramos que pertenece a "EuroNet s.c. Jacek Majak, Aleksandra Kuc" situado en Polonia y cuyo email es "biuro@euronet.net.pl". En el apartado de análisis forense se realizará un análisis más exhaustivo de los datos obtenidos.

Por último, en los C.2.1.Gráficos Resultados 11, se puede observar a qué países pertenecen las direcciones IP de los atacantes. Como se ha visto antes, hay una mayoría que pertenecen a Polonia, además de a otros países como Argentina o Rusia.

5.1.2.2 Análisis estadístico Glastopf 1 y 2

En este caso la diferencia del número de conexiones entre los dos honeypots es abismal. Glastopf 1 ha llegado a 2.341.505 conexiones mientras que Glastopf 2 únicamente a 68, como se puede ver en C.2.2.Gráficos Resultados 14. Además, dichas direcciones provienen

la mayoría de Estados Unidos (C.2.2.Gráficos Resultados 15) y como se ve en C.2.2.Gráficos Resultados 16, son las direcciones que empiezan por "104.192.3.*" que se analizaron en el apartado anterior y siguen realizando conexiones a Glastopf 1.

También será interesante analizar las direcciones que empiezan por "66.249.69.*" pues componen un gran número de las conexiones. Por otra parte, que Glastopf 2 recibiera tan pocas conexiones (C.2.2.Gráficos Resultados 17) es curioso pues los otros honeypots recibieron al menos mil conexiones cada uno aun estando recién desplegados.

Por último, se observa que la mayoría de conexiones a Glastopf siguen siendo de tipo HEAD (C.2.2.Gráficos Resultados 18) provocadas por las direcciones ya mencionadas. Algunos ejemplos de estas conexiones se pueden ver en C.2.2.Gráficos Resultados 19 y Gráficos Resultados 20 que más tarde se analizarán en el apartado de análisis forense.

5.1.2.3 Análisis estadístico Dionaea 1 y 2

Si se observa los C.2.3.Gráficos Resultados 21, se puede comprobar que el número de ataques recibidos y su distribución en el tiempo es muy similar en las dos máquinas que ejecutan Dionaea, llegando a alcanzar 1146 ataques la primera y 1130 la segunda.

Si nos fijamos en las direcciones IP que se han conectado a las máquinas como muestran los C.2.3.Gráficos Resultados 24 y C.2.3.Gráficos Resultados 25 se observa que hay direcciones comunes para las dos máquinas como la "45.32.21.116". En la sección de análisis forense se analizarán algunas de estas direcciones.

También hay que destacar que la mayoría de las direcciones provienen de los mismos países en las dos máquinas (C.2.3.Gráficos Resultados 22). Sin embargo, en el tipo de protocolos de las conexiones difieren un poco (C.2.3.Gráficos Resultados 23), teniendo Dionaea 1 alrededor de cuatro veces más conexiones HTTP que Dionaea 2. Puede ser debido a que Dionaea 1 ha estado más tiempo funcional.

5.1.2.4 Análisis Forense

Glastopf: como ya vimos, en Glastopf continúan los ataques provenientes de las direcciones "104.192.3.*" que ya fueron analizadas y ahora nos enfocaremos en analizar las direcciones "66.249.69.*".

Si buscamos en Shodan y Cymon no obtenemos resultados, pero en AbuseIPDB se nos muestra que pertenecen a Google y es un bot de crawling o araña. La funcionalidad de este tipo de bots es la de encontrar páginas web, escanearlas y volver a rastrear a través de los enlaces que tengan esas páginas web y así propagarse. Además, el bot de Google permite que las páginas web se indexen dentro del navegador de Google gracias a la información que recolecta.

Por otra parte, dicho bot puede ser bloqueado para que no rastree nuestra web en caso de que no queramos ser indexados. Añadir que también existen ataques que utilizan IPs o dominios falsos como los de los bots de Google (ej. "crawl-66-249-66-1.googlebot.com") pero que pueden ser comprobados si pertenecen a Google o no preguntándole al DNS.

Cowrie: para Cowrie analizaremos la dirección "195.22.127.83" pues era común para las dos máquinas. Los resultados de Shodan indican que tiene un servidor SSH en el puerto 22, que es de Polonia y que pertenece a la misma compañía que su ISP "EuroNet s.c. Jacek Majak, Aleksandra Kuc" por lo tanto podemos concluir que es un router particular y no de una compañía.

Si observamos los resultados de Cymon y AbuseIPDB comprobamos que también ha sido reportado por ataques SSH a otros servidores, concretamente ataques de fuerza bruta. Una contramedida sería la de bloquear el protocolo SSH para esta dirección IP o bloquear toda la comunicación con dicha IP.

Dionaea: primeramente, veamos que a Dionaea también han accedido los bots de Google como en Glastopf, pero en mucha menor medida, esto puede ser debido a que Glastopf genera una página aleatoria con cada acceso y si los bots detectan cambios en una página web vuelven a escanearla.

Otra de las IPs con más conexiones es la de "185.38.148.238" al igual que otras IPs analizadas aparece en listas negras o tiene reportes por escaneo de puertos o intentos de ataque por fuerza bruta por SSH. Sin embargo, pertenece a un servidor de una compañía que alquila servidores situada en Bristol, Reino Unido. Esto puede suponer una ventaja y una desventaja a la vez.
Que acceda a través de un servidor alquilado significa que el proveedor de ese servidor conoce los datos del atacante, como, por ejemplo, su tarjeta de crédito. Sin embargo, la empresa no va a entregar los datos de un cliente a no ser que haya realizado un ataque muy serio y lo solicite una autoridad mayor como pueda ser el gobierno de Reino Unido.

5.1.3 Análisis capturas – dentro y fuera del Firewall

Esta etapa abarca desde el 9 de junio a las 12:00 PM hasta el FECHA, cuando los honeypots Cowrie 2, Glastopf 2 y Dionaea 2 se pusieron delante del Firewall de la UAM para comprobar la efectividad de este.

5.1.3.1 Análisis estadístico Cowrie 1 y 2

Como se puede apreciar en el C.3.1.Gráficos Resultados 26, el número de conexiones a Cowrie 2, el que se encuentra sin protección del Firewall, aumentando llegando a tener casi el doble de conexiones que Cowrie 1.

Las dos direcciones IP con más conexiones: "5.188.10.249" y "195.22.127.83" aparecen en ambos honeypots (C.3.1.Gráficos Resultados 28 y C.3.1.Gráficos Resultados 29). El número de conexiones de la dirección "5.188.10.249" aumenta en Cowrie 2 pues no tiene la protección del Firewall y es un resultado esperable. Sin embargo, la dirección "195.22.127.83", ha realizado más conexiones sobre el honeypot protegido por el Firewall.

Otra de las similitudes de ambos honeypots es el top 4 de países atacantes, formado por Rusia, China, Argentina y Polonia como muestra el C.3.1.Gráficos Resultados 27.
Algo que ha llamado nuestra atención son algunos de los usuarios y contraseñas que se han utilizado para acceder por SSH a Cowrie. Como se puede ver en el Gráficos Resultados

30el nombre más utilizado en Cowrie 1 es “mother”, con la contraseña “fucker” algo totalmente inusual y que llama la atención. Este patrón se ha registrado también en Cowrie 2 con 21 conexiones en un único día. En el apartado de análisis forense se muestra la investigación de este caso. Las combinaciones en Glastopf 2 (C.3.1.Gráficos Resultados 31) se mantienen como las típicamente usadas.

5.1.3.2 Análisis estadístico Glastopf 1 y 2

En el caso de Glastopf, el número de ataques han bajado drásticamente con respecto a la fase de control en el honeypot Glastopf 1 que es el que se encuentra bajo la protección del Firewall (C.3.2.Gráficos Resultados 32). Glastopf 2 sigue teniendo muy pocos ataques aun habiendo sido colocado fuera de la protección del Firewall. Puede ser que la IP no esté siendo indexada y por ello los atacantes no la conozcan.

Las direcciones mayoritarias vuelven a repetirse en Glastopf 1, como se puede ver en el C.3.2.Gráficos Resultados 35. En cambio, en Glastopf 2 (C.3.2.Gráficos Resultados 36), hay una nueva dirección que destaca sobre las demás, la “217.61.17.20” con 56 conexiones.
En Glastopf 1, se mantienen los países atacantes y en Glastopf 2 aparece Italia como mayor atacante debido a la dirección “217.61.17.20” (C.3.2.Gráficos Resultados 33 y C.3.2.Gráficos Resultados 34).

Por otro lado, se nos avisó desde el CAU, que tenían una sonda del CERT instalada por encima del Firewall que les avisó de un ataque SQLi por parte de la dirección IP “78.46.174.55” hacia Glastopf 1. En el apartado de análisis forense se detallará los efectos de dicho ataque.

En los C.3.2.Gráficos Resultados 37 y C.3.2.Gráficos Resultados 38 se ven los patrones de las conexiones realizadas durante esta fase. Los patrones más populares en Glastopf 1 vuelven a ser de tipo HEAD y por ello no se muestran de nuevo. En el análisis forense se procederá a analizar parte de estos patrones.

5.1.3.3 Análisis estadístico Dionaea 1 y 2

Como se puede apreciar en el gráfico C.3.3.Gráficos Resultados 39 el número de ataques a Dionaea 2 se ha duplicado con respecto a la fase anterior y con su clon, Dionaea 1, en esta fase. Debido a que el día 20 de junio se cayó una de las máquinas virtuales no se ha tenido en cuenta para el análisis de ambas.

Las direcciones IP con más accesos han sido “95.108.213.25” y “213.32.7.73” en Dionaea 1 y 2 respectivamente (C.3.3.Gráficos Resultados 42 y C.3.3.Gráficos Resultados 43). La primera es una dirección rusa que es la que ha generado la mayoría de conexiones en Dionaea 1 de ese país (C.3.3.Gráficos Resultados 40). La segunda es de Estados Unidos y curiosamente ninguna de las dos ha sido reportada anteriormente ni figuran en listas negras.

Además, gran mayoría de estos ataques nuevos son del tipo “mssql” como se ven en C.3.3.Gráficos Resultados 41, que si recordamos es el servicio dedicado a simular un

servidor SQL de Microsoft y que previamente no había sido atacado. Se nos comunicó que todos los ataques de este tipo estaban siendo bloqueados por el Firewall de la UAM y es por ello que no se habían registrado previamente.

Otros servicios en los que ha aumentado la cantidad de ataques registrados son "upnpd" y "SipSession". "SipSession" corresponde al protocolo SIP (*Session Initiation Protocol*) [55], cuya función es la interacción de dispositivos a través de sesiones para enviar contenido multimedia; está basado en HTTP y SMTP.

Por otro lado, "upnpd" corresponde al protocolo UPnP (Universal Plug and Play) [56] es un conjunto de protocolos de comunicación que permite a periféricos en red, como computadoras personales, impresoras, puntos de acceso Wi-Fi y dispositivos móviles, descubrir la presencia de otros dispositivos en la red y establecer servicios de red de comunicación.

En el análisis forense se detallará mejor que tipos de ataques se pueden realizar con estos protocolos.

5.1.3.4 Análisis Forense

Cowrie: en total se encontraron tres direcciones IP que utilizaban este usuario y que siempre realizaban lo mismo: probar con dicho usuario y contraseña durante 21 veces exactamente.
Las dos hipótesis que se plantearon fueron: dichas IPs pertenencen a una botnet o podría ser alguien que ha detectado que el sistema al que está entrando es un Honeypot utilizando distintas IPs.

Para saber si pertenecían a una botnet conocida se utilizó MyIP.ms [57], con resultados negativos. Sin embargo, se comprobó que las tres habían sido añadidas a listas negras por realizar ataques de fuerza bruta por SSH. Por ello se cree que puede ser el mismo atacante usando distintas IPs.

Glastopf: primero analizaremos el ataque que detectó el sensor del CERT con la dirección "78.46.174.55".
Gracias a Shodan y AbuseIPDB sabemos que se encuentra en Alemania, tiene el puerto 22 asignado a OpenSSH y ha sido reportado múltiples veces por ataques DDoS, escaneo de puertos e inyección SQL entre otros.

Si analizamos los datos de Glastopf 1, vemos que ha realizado 6.402 conexiones alrededor del día 16 de junio únicamente. De esas conexiones Glastopf únicamente ha reconocido una como inyección SQL, el resto las ha clasificado como login, unkonwn o comments.
Esto puede ser debido a que la inyección SQL realizada es más nueva que la base de datos que Glastopf tiene con el listado de inyecciones conocidas y por eso las clasifica como login.

Algunos de estos patrones que hemos analizado son:

- "GET /login-redirect/index.php?body=": este tipo de patrón corresponde a Wordpress [58]. Consiste en poner en la URL la siguiente cadena y así accede a la ventana donde estaría el usuario logeado. Es decir, intenta saltarse el control de

login y acceder a la cuenta del usuario directamente. Hay multitud de estos patrones, luego el atacante ha ejecutado un programa automático para el escaneo de vulnerabilidades enfocadas a Wordpress.

- "GET /login_admin/8003/test-cgi": intenta accede al script "test-cgi" que tienen algunos servidores HTTP. Este script permite listar todos los directorios del servidor y debería ser eliminado en un servidor operativo.
- "GET /NSearch/index.php?body=": en este caso intenta acceder a los directorios de una herramienta específica, Nsearch [59]. Esta herramienta es un buscador de scripts para Nmap. Nmap ya tiene unos scripts por defecto, pero se pueden crear nuevos scripts personalizados si se encuentran nuevas vulnerabilidades y así buscarlas en los siguientes escaneos. Al atacante le permite ampliar su base de datos de vulnerabilidades conocidas en caso de hacerse con estos scripts.
- También ha habido intentos de Inyección SQL directa, pero Glastopf los almacena codificados y se necesita un decodificador de URL para poder leerlos cómodamente. Algunos de estos ataques como "/(select extractvalue(xmltype(' " consisten en sacar información de nuestra base de datos SQL utilizando las malas configuraciones que sufren los campos de los formularios PHP.

En conclusión, este usuario ha hecho un análisis de vulnerabilidades automático sobre nuestro honeypot.

Dionaea: en esta fase hemos visto que atacan nuevos servicios que detrás del Firewall no atacaban. Podemos ver las acciones que realizan los atacantes y replicarlas con los archivos que genera Dionaea en la carpeta bistreams. Para ejecutarlos Dionaea dispone de un script en la carpeta "`/opt/dionaea/modules/python/retry.py`". Algunos de los ejemplos de los ataques:

- MSSQL: Los atacantes acceden a la supuesta base de datos como si tuvieran todos los privilegios. Una vez dentro algunos de ellos intentan robar la base de datos creando transacciones que después eliminan, crear objetos OLE [60], o eliminar "jobs" o trabajos como "task.exe" o "regs.exe".
- SIP: en la mayoría de los casos se inicia una sesión SIP y después se cierra la sesión, probablemente sea debido a que es un escaneo.
- UPnP: todas las conexiones únicamente escanean por dispositivos UPnP con M-SEARCH [61]. Dionaea devuelve la localización del XML con la información y los servicios que ofrece (http://IP_DIONAEA:49152/IPMIdevicedesc.xml) pero si accedemos a él a través del navegador nos salta error de conexión.

5.1.4 Conclusión del análisis del Firewall

Después de la realización del análisis se ha podido observar que el Firewall de la UAM detiene bastantes ataques, pero aun así siguen llegando a nuestros honeypots.

Algunos de los servicios como UPnP, SIP o MSSQL son directamente bloqueados por el Firewall de la UAM y es por ello que no aparecían ataques relacionados con ellos cuando estaban todos los honeypots protegidos por el Firewall.

Por otra parte, Glastopf 1 recibió ataques de inyección SQL que el Firewall no llegó a detectar o filtrar y Glastopf 2 no aumentó su número de conexiones como ocurrió con Dionaea 2 o Cowrie 2 al quitarles el Firewall. Puede ser debido a que es un dominio reciente.

En Cowrie sí que hubo un incremento de conexiones al quitarle la protección del Firewall, debido a que el Firewall filtra direcciones IP basándose en listas negras. El problema de las conexiones SSH es que no sabes si al realizar una conexión es legítima o no sólo con conectarte al servidor, sino que es necesario analizar los mensajes posteriores.

Por último, durante el análisis nunca hubo un registro de ataque DDoS que tirase nuestros servidores. Esto es debido a que la UAM bloquea cualquiera de sus sub-dominios si ve que estos están siendo víctimas de un ataque de denegación de servicio. Esto afecta a todos los dominios independientemente de si están delante o detrás del Firewall.

En la Tabla 5-1, se muestran el número total de conexiones que han accedido a los honeypots antes de la realización del análisis del Firewall y durante este.

Honeypot	E. Previa	Antes del Firewall	Después del Firewall	Total
Cowrie 1	7.881	220	1.226	9.327
Cowrie 2	-	206	2.053	2.259
Glastopf 1	18.725.018	2.341.505	1.654.437	22.720.960
Glastopf 2	-	68	217	285
Dionaea 1	45.864	1146	7.903	54.913
Dionaea 2	-	1130	10.624	11.754

Tabla 5-1: Conexiones Totales en los Honeypots

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Durante este trabajo se ha demostrado el potencial que tienen los honeypots en el ámbito de la seguridad informática. Se han estudiado varios de los honeypots actuales y realizado pruebas de pentesting para posteriormente configurarlos y que no fuesen detectados como sistemas falsos por los atacantes.

Se ha instalado y configurado Kibana para poder visualizar los ataques recibidos por los honeypots de una forma rápida y realizar análisis forenses de una forma más eficaz. Gracias a la implementación de Filebeat en las máquinas virtuales podemos agregar nuevos sensores de forma automática y visualizarlos en Kibana.

Respecto al Firewall de la UAM, se ha demostrado su gran efectividad. Sin embargo, aún llegan algunos ataques que el Firewall no es capaz de bloquear. Gracias a las capturas de los honeypots se podría aumentar su seguridad continuando la colaboración con el CAU, al cual quiero agradecer su ayuda y colaboración durante el desarrollo del proyecto en especial a D. Carlos Ramírez, por atendernos tan atentamente.

En referencia a Glastopf 2, no se sabe aún por qué es tan bajo el número de ataques en dicho honeypot. Suponemos que es debido a que la IP de Glastopf 2 es nueva y no tiene indexación en Google, y por ello no acceden los atacantes. Si lo comparamos con el proyecto que realizó D. Diego Jurado Pallarés [11], el cual ha servido de inspiración, hemos recibido muy pocos ataques en Glastopf 2.

Por último, remarcar que gracias a la creación de la máquina virtual distribuible podremos agregar nuevos sensores situados en distintas subredes y comprobar la eficacia de los distintos anillos de seguridad que forman la red de la UAM como de otras redes.

6.2 Trabajo futuro

Una vez concluido el proyecto se proponen algunas ideas para mejorarlo en un futuro:

- Crear un sistema de usuarios para que únicamente puedan acceder los usuarios registrados a los datos recogidos por todos los honeypots o que hayan dejado la máquina virtual distribuible funcionando durante cierto tiempo.
- Crear un algoritmo pseudo-aleatorio que genere una nueva configuración en la máquina virtual distribuible con cada descarga y que a la vez siga generando una apariencia realista para los atacantes.
- Crear un sistema de reglas dinámicas para un Firewall que varíe según los resultados obtenidos por los honeypots.
- Añadir nuevos tipos de honeypots que no han sido incluidos en este proyecto.
- Crear una web que publique blacklists con las direcciones IP que han atacado a nuestros honeypots y así otros usuarios sean advertidos.
- Generar alertas a partir de los datos recibidos por los honeypots y así tener un sistema con un tiempo de reacción mayor

Referencias

[1] EL PAÍS, "Los ciberataques en España se duplican en un año: más de 105.000 en 2016," p. 1, 14-Mar-2017.

[2] Steve Morgan, "Top 2016 Cybersecurity Reports Out From AT&T, Cisco, Dell, Google, IBM, McAfee, Symantec And Verizon," *Forbes*, p. 1, 2016.

[3] E. Pérez López, "SCADA systems in the industrial automation," *Tecnol. en Marcha*, vol. 28, no. 4, pp. 3–14, 2015.

[4] M. del Interior, "Centro Nacional de Protección de Infraestructuras Críticas CNPIC." [Online]. Available: http://www.cnpic.es/. [Accessed: 20-Mar-2017].

[5] R. C. Joshi and A. Saranda, *Honeypots: A New Paradigm to Information Security*, vol. 1, no. 11. 2011.

[6] B. Cheswick and AT&T Bell Laboratories, "An Evening with Berferd In Which a Cracker is Lured, Endured, and Studied," *IEEE Secur. Priv.*, vol. 11, no. 2, pp. 47–54, 1992.

[7] "HoneyNet Project - Old." [Online]. Available: http://old.honeynet.org/misc/project.html. [Accessed: 24-Jan-2017].

[8] "HoneyNet Project." [Online]. Available: http://www.honeynet.org/about. [Accessed: 24-Jan-2017].

[9] Cisco, "¿Qué es un firewall?" [Online]. Available: http://www.cisco.com/c/es_es/products/security/firewalls/what-is-a-firewall.html. [Accessed: 15-Apr-2017].

[10] C. R. Moral, "Detectando y Ocultando Honeypots: Hidden-Pot," Universidad Autónoma de Madrid (UAM), 2015.

[11] D. J. Pallarés, "Análisis y Estudio de Honeypots Complejos: Honeynets," Universidad Autónoma de Madrid (UAM), 2016.

[12] D. M. Trujillano, "Sistema adaptativo de prevención de intrusos mediante Honeypots," Universidad Autónoma de Madrid (UAM), 2016.

[13] J. M. F. Marín, "Virtual Honeynets," 2013.

[14] R. Kamboj and V. Rana, "Implementation of Attack Data Collection Incorporating Multi Level Detection Capabilities Using Low Interaction Honeypot," *Int. J. Comput. Sci. Eng.*, vol. 2, no. 4, pp. 27–36, 2013.

[15] Y. Birdi, "A Review of Honey Net Technology," *Int. J. Comput. Sci. Commun. Eng.*, pp. 100–103, 2013.

[16] L. Spitzner, "To Build a Honeypot," 1999. [Online]. Available: http://www.scn.rain.com/~neighorn/docs/security/misc/honeypot.html. [Accessed: 26-Jun-2017].

[17] L. Spitzner, "Honeypot Farms," 2003. [Online]. Available: https://www.symantec.com/connect/articles/honeypot-farms. [Accessed: 26-Jun-2017].

[18] F. Cócaro, M. G. María, and J. Rouiller, "Diseño E Implantación De Un Honeypot," *InCo – Fac. Ing. -Universidad la República*, 2007.

[19] M. Oosterhof, "Kippo - Github." [Online]. Available: https://github.com/desaster/kippo. [Accessed: 26-Jun-2017].

[20] J. A. Coret, "Kojoney - A Honeypot For The SSH Service." [Online]. Available: http://kojoney.sourceforge.net/. [Accessed: 05-May-2017].

[21] InfoSec, "Tracking Attackers with a Honeypot – Part 2 (Kippo)." [Online]. Available: http://resources.infosecinstitute.com/tracking-attackers-honeypot-part-2-

kippo/#gref. [Accessed: 05-Mar-2017].
[22] M. Oosterhof, “Cowrie - GitHub.” [Online]. Available: https://github.com/micheloosterhof/cowrie. [Accessed: 07-May-2017].
[23] DinoTools, “Dionaea - GitHub.” [Online]. Available: https://github.com/DinoTools/dionaea. [Accessed: 20-Jan-2017].
[24] E. Tan, “Dionaea – A Malware Capturing Honeypot.” [Online]. Available: https://www.edgis-security.org/honeypot/dionaea/. [Accessed: 13-Feb-2017].
[25] Adnan Mohd Shukor, “LibEmu Library.” [Online]. Available: https://launchpad.net/libemu. [Accessed: 20-Mar-2017].
[26] L. “glaslos” Rist, J. “johnny” Vestergaard, D. “creo” Haslinger, and A. “De” Pasquale, “MushMush Foundation.” [Online]. Available: http://mushmush.org/. [Accessed: 20-Feb-2017].
[27] M. Foundation, “CONPOT ICS/SCADA Honeypot.” [Online]. Available: http://conpot.org/. [Accessed: 10-Jan-2017].
[28] N. Instruments, “Información Detallada sobre el Protocolo Modbus.” [Online]. Available: http://www.ni.com/white-paper/52134/es/. [Accessed: 26-Jun-2017].
[29] C. Gonzalez, “What are Human Machine Interfaces and Why Are They Becoming More Important?” [Online]. Available: http://beta.machinedesign.com/iot/what-are-human-machine-interfaces-and-why-are-they-becoming-more-important?utm_test=redirect&utm_referrer=https%3A%2F%2Fwww.google.es%2F. [Accessed: 13-Feb-2017].
[30] C. S. Vetsch, M. Koßin, and M. Mauer, “Know Your Tools : Glastopf,” *Honeynet Proj.*, pp. 1–29, 2010.
[31] M. Foundation, “Glastopf - GitHub.” [Online]. Available: https://github.com/mushorg/glastopf. [Accessed: 26-Jun-2017].
[32] M. Schloesser, “HPFeeds - GitHub.” [Online]. Available: https://github.com/rep/hpfeeds. [Accessed: 26-Jun-2017].
[33] M. Foundation, “SNARE - GitHub.” [Online]. Available: https://github.com/mushorg/snare. [Accessed: 26-Jun-2017].
[34] M. Foundation, “TANNER - GitHub.” [Online]. Available: https://github.com/mushorg/tanner. [Accessed: 26-Jun-2017].
[35] Lanjelot, “Twisted-Honeypots - GitHub.” [Online]. Available: https://github.com/lanjelot/twisted-honeypots. [Accessed: 23-May-2017].
[36] T. M. Labs, “Twisted Engine.” [Online]. Available: http://twistedmatrix.com/trac/. [Accessed: 26-Jun-2017].
[37] J. Broad and A. Bindner, *Hacking with Kali*. 2014.
[38] Shodan®, “Shodan - Webpage.” [Online]. Available: https://www.shodan.io/. [Accessed: 20-Feb-2017].
[39] E. Paths, “FOCA.” [Online]. Available: https://www.elevenpaths.com/es/labstools/foca-2/index.html. [Accessed: 20-Feb-2017].
[40] Shodan®, “Honeypot Or Not?” [Online]. Available: https://honeyscore.shodan.io/. [Accessed: 20-Feb-2017].
[41] Gordon Lyon, “Nmap: the Network Mapper.” [Online]. Available: https://nmap.org/. [Accessed: 20-Feb-2017].
[42] Elastic, “The Open Source Elastic Stack.” [Online]. Available: https://www.elastic.co/. [Accessed: 22-May-2017].
[43] T. P. G. D. Group, “PostgreSQL - Webpage.” [Online]. Available: https://www.postgresql.org/. [Accessed: 20-Feb-2017].
[44] ORACLE, “VirtualBox - Webpage.” [Online]. Available:

https://www.virtualbox.org/. [Accessed: 09-Jan-2017].
[45] SPI, “Debian - Webpage.” [Online]. Available: https://www.debian.org/index.es.html. [Accessed: 28-Apr-2017].
[46] C. L. Ubuntu, “Ubuntu - Webpage.” [Online]. Available: https://www.ubuntu.com/. [Accessed: 16-Feb-2017].
[47] R. Espadas, “DionaeaFR - Github.” [Online]. Available: https://github.com/rubenespadas/DionaeaFR. [Accessed: 26-May-2017].
[48] D. S. Foundation, “Django.” [Online]. Available: https://www.djangoproject.com/. [Accessed: 12-May-2017].
[49] M. P. Esteso, “PROGRAMAR TAREAS EN LINUX USANDO CRONTAB.” [Online]. Available: https://geekytheory.com/programar-tareas-en-linux-usando-crontab. [Accessed: 02-Jun-2017].
[50] A. Lin, “Elasticsearch Architectural Overview.” [Online]. Available: https://buildingvts.com/elasticsearch-architectural-overview-a35d3910e515. [Accessed: 27-Jun-2017].
[51] A. Vanderbush, “How to Export Data from Elasticsearch into a CSV File.” [Online]. Available: https://qbox.io/blog/how-to-export-data-elasticsearch-into-csv-file. [Accessed: 03-Jun-2017].
[52] K. Boeckman, “Docker containers vs. virtual machines: What’s the difference?” [Online]. Available: https://newsroom.netapp.com/blogs/containers-vs-vms/. [Accessed: 26-Jun-2017].
[53] ESentire, “Cymon.io.” [Online]. Available: https://cymon.io/. [Accessed: 02-Mar-2017].
[54] DigitalOcean, “AbuseIPDB.”
[55] 3CX, “Informacion y Preguntas frecuentes acerca de SIP.” [Online]. Available: https://www.3cx.es/voip-sip/sip-faq/. [Accessed: 27-Jun-2017].
[56] R. Velasco, “UPnP, el protocolo que puede saltarse la seguridad de tu Router o Firewall.” [Online]. Available: https://www.redeszone.net/2015/09/15/upnp-el-protocolo-que-puede-saltarse-la-seguridad-de-tu-router-o-firewall/. [Accessed: 26-Jun-2017].
[57] Myip.ms, “Myip.ms.” [Online]. Available: http://bot.myip.ms/. [Accessed: 22-Jun-2017].
[58] Automattic, “Wordpress.” [Online]. Available: https://es.wordpress.com/. [Accessed: 27-Jun-2017].
[59] DragonJAR, “NSEarch (Nmap Script Engine Search).” [Online]. Available: NSEarch (Nmap Script Engine Search). [Accessed: 27-Jun-2017].
[60] Microsoft, “Crear, modificar y controlar objetos OLE.” [Online]. Available: https://support.office.com/es-es/article/Crear-modificar-y-controlar-objetos-OLE-e73867b2-2988-4116-8d85-f5769ea435ba. [Accessed: 12-Jun-2017].
[61] S. Lopez, “Writing a UPnP Control Point in JavaScript, Part Two.” [Online]. Available: https://sethlopez.me/article/writing-a-upnp-control-point-in-javascript-part-two/. [Accessed: 26-Jun-2017].
[62] VANIMPE, “Install DionaeaFR web frontend to Dionaea honeypot on Ubuntu.” [Online]. Available: https://www.vanimpe.eu/2014/07/04/install-dionaeafr-web-frontend-dionaea-ubuntu/. [Accessed: 13-May-2017].
[63] J. Ellingwood, “How To Install Elasticsearch, Logstash, and Kibana (ELK Stack) on Ubuntu 16.04.” [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elk-stack-on-ubuntu-16-04. [Accessed: 06-Feb-2017].
[64] Cudeso, “cudeso-honeypot.” [Online]. Available: https://github.com/cudeso/cudeso-

honeypot. [Accessed: 24-Jun-2017].
[65] Massachusetts Institute of Technology, “Los wrappers TCP y el comando xinetd.” [Online]. Available: http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-tcpwrappers.html. [Accessed: 13-Jun-2017].
[66] Microsoft, “Boletín de seguridad de Microsoft MS10-061 - Crítica.” [Online]. Available: https://technet.microsoft.com/es-es/library/security/ms10-061.aspx. [Accessed: 12-Mar-2017].
[67] B. Lab, “HoneyDrive.” [Online]. Available: http://bruteforcelab.com/honeydrive. [Accessed: 29-Jun-2017].
[68] U. Network, “How to Import/Export OVA Files in VirtualBox.” [Online]. Available: https://www.maketecheasier.com/import-export-ova-files-in-virtualbox/. [Accessed: 01-Jun-2017].

Glosario

Ataque zero-day	Es un ataque hacia un sistema que no se había registrado antes y por tanto los sistemas de defensa no lo reconocen.
CNPIC	Centro Nacional de Protección de Infraestructuras Críticas
DionaeaFR	Servidor Front-End desarrollado por Rubén Espadas para la visualización de la base de datos de Dionaea.
Dork (de Google)	Los Google dorks son combinaciones de operadores de búsqueda especiales que se utilizan para extraer información valiosa o sensible desde Google.
ELK Stack	Arquitectura Elasticsearch, Logstash, Kibana (y ocasionalmente Filebeat)
Elasticsearch	Un motor de búsqueda basado en Lucene.
Firewall	Es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas
Filebeat	Aplicación encargada de leer y reenviar *logs* a otro servidor.
FOCA	*Fingerprinting Organizations with Collected Archives*, es una herramienta utilizada principalmente para encontrar metadatos e información oculta en los documentos que examina
HIDS	*Host-based intrusion detection system*, busca detectar anomalías que indican un riesgo potencial, revisando las actividades en la máquina (host)
HPFeeds	Es un protocolo ligero con autentificación con un modelo publicador-suscriptor.
IDS	*Intrusion Detection System*, es un programa de detección de accesos no autorizados a un computador o a una red
Kali Linux	Distribución Linux con herramientas enfocadas al hacking ético.
Kibana	Es un plugin de código abierto para la visualización de los datos de Elasticsearch
LibEmu	Librería que sirve para emular una consola x86
Logstash	Es una herramienta para la administración de *logs*. Esta herramienta se puede utilizar para recolectar, parsear y guardar los *logs* para futuras búsquedas
MSSQL	Es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft.
Nginx	Es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico.
NIDS	*Network Intrusion Detection System*. Busca detectar anomalías que inicien un riesgo potencial, tales como ataques de denegación de

	servicio, escaneadores de puertos o intentos de entrar en un ordenador, analizando el tráfico en la red en tiempo real
Nmap	Es un programa de código abierto que sirve para efectuar rastreo de puertos.
Pentesting	Una prueba de penetración, o "*pentest*", es un ataque a un sistema informático con la intención de encontrar las debilidades de seguridad y todo lo que podría tener acceso a ella, su funcionalidad y datos
PostgreSQL	Es un sistema de gestión de bases de datos relacional orientado a objetos y libre.
Proxy	Es un servidor que hace de intermediario en las peticiones de recursos que realiza un cliente a otro servidor
SCADA	*Supervisory Control And Data Acquisition*, es un concepto que se emplea para realizar un software para ordenadores que permite controlar y supervisar procesos industriales a distancia
Shodan	Es un motor de búsqueda que le permite al usuario encontrar iguales o diferentes tipos específicos de equipos (routers, servidores, etc.) conectados a Internet a través de una variedad de filtros
SIEM	*Security information and event management,* productos software que proveen de un análisis en tiempo real de las alertas de seguridad generadas por aplicaciones de la red
SIP	*Session Initiation Protocol,* es un protocolo está para la iniciación, modificación y finalización de sesiones interactivas de usuario donde intervienen elementos multimedia como el video, voz, mensajería instantánea, juegos en línea y realidad virtual.
SMB	*Server Message Block,* es un protocolo de red que permite compartir archivos, impresoras, etcétera, entre nodos de una red de computadoras que usan el sistema operativo Microsoft Windows
SMTP	*Simple Mail Transfer Protocol*, es un protocolo de red utilizado para el intercambio de mensajes de correo electrónico
SQL Injection	Inyección SQL, es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos
SSL	*Secure Sockets Layer*, protocolo criptográfico anterior a TLS
TFTP	Trivial file transfer Protocol, es un protocolo de transferencia muy simple semejante a una versión básica de FTP
TLS	*Transport Layer Security*, protocolo criptográfico que usa criptografía asimétrica con certificados X.509
VirtualBox	Es un software de virtualización para arquitecturas x86/amd64, creado originalmente por la empresa alemana innotek Gmb y actualmente desarrollado por Oracle

7 Anexos

A. Manual de instalación

A.1 Instalación Honeypot Dionaea

A.1.1 Prerrequisitos

Actualizamos el sistema:

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
```

Instalamos paquetes básicos:

```
$ sudo apt-get install software-properties-common
```

Instalamos los prerrequisitos necesarios de Ubuntu 14.04:

```
$ sudo aptitude -f install libudns-dev libglib2.0-0 libglib2.0-dev libssl-
dev libcurl4-openssl-dev libreadline-dev sqlite sqlite3 libsqlite3-dev
python-dev libtool automake autoconf build-essential  subversion git-core
flex bison pkg-config sqlite3 libgc-dev libev4 libemu2 libemu-dev
python3.4 python3-all readline-common cython cython3 curl pcap libpcap0.8
libnl1 libnl-3-200
```

Instalamos Ilbcfg:

```
$ cd /opt/
$ git clone  git://git.carnivore.it/liblcfg.git liblcfg
//Alternativa: https://github.com/ThomasAdam/liblcfg
$ cd liblcfg/code
$ autoreconf -f -i -Wall,no-obsolete
$ ./configure --prefix=/opt/dionaea
$ make install
```

Instalamos LibEmu:

```
$ cd /opt/
$ git clone https://github.com/buffer/libemu
$ cd libemu
$ sudo autoreconf -v -i
$ sudo ./configure --prefix=/opt/dionaea; make install
$ make install
```

Instalamos (opcionalmente) LibEv de forma manual:

```
$ cd /opt/
$ sudo wget http://dist.schmorp.de/libev/libev-4.23.tar.gz
$ tar xfz libev-4.23.tar.gz
$ cd libev-4.23
$ sudo ./configure --prefix=/opt/dionaea
```

```
$ sudo make install
```

Instalamos LibPcap:

```
$ cd /opt/
$ sudo wget http://www.tcpdump.org/release/libpcap-1.1.1.tar.gz
$ sudo tar -xvzf libpcap-1.1.1.tar.gz
$ cd libpcap-1.1.1
$ sudo ./configure --prefix=/opt/dionaea
$ sudo make
$ sudo make install
```

Instalamos Python:

```
$ cd /opt/
$ sudo wget http://www.python.org/ftp/python/3.2.2/Python-3.2.2.tgz
$ sudo tar -xvzf Python-3.2.2.tgz
$ cd Python-3.2.2
$ sudo ./configure --enable-shared --prefix=/opt/dionaea --with-computed-
gotos --enable-ipv6 LDFLAGS="-Wl,-rpath=/opt/dionaea/lib/ -
L/usr/lib/x86_64-linux-gnu/"
$ make
$ make install
```

Instalamos Cython:

```
$ cd /opt/
$ sudo wget http://cython.org/release/Cython-0.25.2.tar.gz
$ //Alternativa // sudo git clone https://github.com/cython/cython
$ sudo tar -xvzf Cython-0.16.tar.gz
$ cd Cython-0.16
$ sudo python setup.py install
```

A.1.2 Instalación

Creamos un usuario nuevo llamado “dionaea”:

```
$ sudo useradd dionaea
$ sudo groupadd dionaea
$ sudo usermod dionaea -G dionaea
$ sudo chown -R dionaea:dionaea /opt/dionaea/var/log/
```

Ahora tenemos 2 caminos a proceder:

Opción A, instalación automática (recomendado)

Añadimos el repositorio de PPA:

```
$ sudo add-apt-repository ppa:honeynet/nightly
$ sudo apt-get update
```

Instalamos el paquete por apt:

```
$ sudo apt-get install Dionaea
```

Y ahora bastaría con iniciar el servicio:

```
$ sudo service dionaea start
```

Opción B, instalación manual

Creamos una carpeta en opt:

```
$ sudo mkdir /opt/dionaea
$ cd /opt/dionaea
```

Descargamos la versión de Dionaea de Dinotools (más actualizada):

```
$ git clone https://github.com/DinoTools/dionaea
```

Instalamos Dionaea:

```
$ sudo autoreconf -v -i
$ sudo ./configure \
-with-lcfg-include=/opt/dionaea/include/ \
-with-lcfg-lib=/opt/dionaea/lib/ \
-with-python=/opt/Python-3.2.2 \
-with-cython-dir=/opt/dionaea/bin \
-with-udns-include=/opt/dionaea/include/ \
-with-udns-lib=/opt/dionaea/lib/ \
-with-emu-include=/opt/dionaea/include/ \
-with-emu-lib=/opt/dionaea/lib/ \
-with-gc-include=/usr/include/gc \
-with-ev-include=/opt/dionaea/include \
-with-ev-lib=/opt/dionaea/lib \
-with-nl-include=/opt/dionaea/include \
-with-nl-lib=/opt/dionaea/lib/ \
-with-curl-config=/usr/bin/ \
-with-pcap-include=/opt/dionaea/include \
-with-pcap-lib=/opt/dionaea/lib/

$ sudo make
$ sudo make install
```

Y ahora bastaría con iniciar el servicio:

```
$ sudo service dionaea start
```

**Adicionales:
Instalar jinja2 para usar templates:

```
$ pip install Jinja2
```

Comprobar que el puerto 80 lo escucha Dionaea y no Apache

```
$ sudo netstat -luntp | grep 80
$ sudo pkill apache2
$ sudo service dionaea restart
$ sudo netstat -luntp | grep dionaea
```

A.2 Instalación Honeypot Kippo

A.2.1 Prerrequisitos

Instalamos los paquetes necesarios:

```
$ apt-get install python-dev openssl python-openssl python-pyasn1 python-twisted
```

Si queremos cambiar el puerto donde irá Kippo (por defecto es el 22 en ssh) lo cambiamos en el siguiente documento:

```
$ sudo nano /etc/ssh/sshd_config
```

Y reiniciamos el servidor ssh:

```
$ sudo /etc/init.d/ssh restart
```

Ahora creamos un nuevo usuario para Kippo:

```
$ adduser kippo
```

Y le damos permisos de sudo:

```
$ visudo
```

Agregando la siguiente línea al documento:

```
kippo ALL=(ALL:ALL) ALL
```

Ahora preparamos el puerto para que lo use Kippo

```
$ touch /etc/authbind/byport/22
$ chown kippo:kippo /etc/authbind/byport/22
$ chmod 777 /etc/authbind/byport/22
```

A.2.2 Instalación

Nos descargamos Kippo:

```
$ git clone https://github.com/desaster/kippo.git
```

Cambiamos el archivo “cfg.dist” a “.cfg” y lo modificamos para que apunte al puerto elegido (en nuestro caso el 22):

```
$ cp kippo.cfg.dist kippo.cfg
$ nano kippo.cfg
```

Y para terminar cambiamos el archivo “start.sh” para que escuche en el puerto 22 también:

```
$ nano start.sh
```

Cambiamos la siguiente línea:

```
twistd -y kippo.tac -l log/kippo.log --pidfile kippo.pid
```

por

```
authbind --deep twistd -y kippo.tac -l log/kippo.log --pidfile kippo.pid
```

Y ejecutamos el honeypot:

```
$ ./start.sh
```

A.3 Instalación Honeypot Cowrie

A.3.1 Prerrequisitos

Comprobamos que los paquetes necesarios están instalados:

```
$ sudo apt-get install git python-dev python-openssl openssh-server
python-pyasn1 python-twisted authbind
```

Cambiamos el puerto del servidor ssh a otro (al 8742 por ejemplo) en el archivo de sshd_config:

```
$ sudo nano /etc/ssh/sshd_config
```

Añadir nuevo usuario cowrie (la pass que queramos y los demás campos en blanco):

```
$ sudo adduser cowrie
```

Preparamos Cowrie para que escuche en el puerto 22:

```
$ sudo touch /etc/authbind/byport/22
$ sudo chown cowrie /etc/authbind/byport/22
$ sudo chmod 777 /etc/authbind/byport/22
```

A.3.2 Instalación

Cambiamos de usuario a Cowrie y nos movemos a /home/cowrie/ donde lo instalaremos:

```
$ su cowrie
$ cd /home/cowrie/
```

Ahora descargamos Cowrie de alguno de estos repositorios (el primero suele estar más actualizado):

```
$ git clone https://github.com/micheloosterhof/cowrie
```

```
//git clone https://github.com/cowrie/cowrie.git
```

Nos movemos a la carpeta Cowrie que se ha generado (/home/cowrie/cowrie)

```
$ cd cowrie
```

Cambiamos el nombre a la configuración

```
$ mv cowrie.cfg.dist cowrie.cfg
```

A continuación, modificamos el archivo de configuración.
1º cambiamos el puerto (NO hay que poner las comillas):

```
"# Port to listen for incoming SSH connections.
#
# (default: 2222)
listen_port = 22"
```

2º cambiamos el nombre de host (por ejemplo 'uamserver'):

```
"# Hostname for the honeypot. Displayed by the shell prompt of the virtual
# environment
#
# (default: svr04)
hostname = uamserver"
```

Ya podemos iniciar start.sh:

```
$ ./start.sh
```

A.4 Instalación Honeypot Glastopf

En el Github de Glastopf [31], podemos obtener las instrucciones de instalación tanto para Debian, como para Ubuntu. En esta guía se mostrará la instalación en Debian.

A.4.1 Prerrequisitos

Añadimos el repositorio de Backports a nuestra lista:

```
$ deb http://backports.debian.org/debian-backports squeeze-backports main
```

Instalamos dependencias:

```
$ apt-get update
$ apt-get install python python-openssl python-gevent libevent-dev python-
dev build-essential make
$ apt-get install python-argparse python-chardet python-requests python-
sqlalchemy python-lxml
$ apt-get install python-beautifulsoup mongodb python-pip python-dev
python-setuptools
$ apt-get install g++ git php5 php5-dev liblapack-dev gfortran
$ apt-get install libxml2-dev libxslt-dev
$ apt-get install libmysqlclient-dev
```

```
$ pip install --upgrade distribute
```

Instalamos el PHP Sanbox:

```
$ cd /opt
$ git clone git://github.com/mushorg/BFR.git
$ cd BFR
$ phpize
$ ./configure --enable-bfr
$ make &&  make install
```

Por último buscamos el path a "bfr.so" y lo copiamos en el archivo de configuración "php.ini":

```
zend_extension =   /usr/lib/php5/20131226/bfr.so
```

A.4.2 Instalación

Existen dos opciones

Opción A:
Instalamos el honeypot a través de pip:

```
$ pip install glastopf
```

Opción B:
Agregamos el Github para obtener las versiones de desarrollo más actualizadas:

```
$ cd /opt
$ git clone https://github.com/mushorg/glastopf.git
$ git clone https://github.com/client9/libinjection.git
$ git clone https://github.com/mushorg/pylibinjection.git
$ cd glastopf
$ python setup.py install
```

Ahora preparamos el entorno de Glastopf:

```
$ cd /opt
$ mkdir myhoneypot
$ cd myhoneypot
$ glastopf-runner
```

Esto crea una nueva configuración en el directorio "/opt/myhoneypot" con el nombre de "glastopf.cfg". La configuración usada para Glastopf es la siguiente:

```
[webserver]
host = 0.0.0.0
#host = 127.0.0.1
port = 80
uid = nobody
gid = nogroup
proxy_enabled = False
```

```
[ssl]
enabled = False
certfile =
keyfile =

[logging]
consolelog_enabled = True
filelog_enabled = True
logfile = log/glastopf.log

[dork-db]
enabled = True
pattern = rfi
mnem_service = False

[hpfeed]
enabled = False
#host = hpfriends.honeycloud.net
#port = 20000
#secret = 3wis3l2u5l7r3cew
#chan_events = glastopf.events
#chan_files = glastopf.files
#ident = x8yer@hp1

[main-database]
enabled = True
#connection_string = sqlite:///db/glastopf.db
connection_string = mysql://glaspot:glaspot@localhost/glaspot

[surfcertids]
enabled = False
host = localhost
port = 5432
user =
password =
database = idsserver

[syslog]
enabled = False
socket = /dev/log

[mail]
enabled = False
patterns = rfi,lfi
user =
pwd =
mail_from =
mail_to =
smtp_host = smtp.gmail.com
smtp_port = 587

[taxii]
enabled = False
host = taxiitest.mitre.org
port = 80
inbox_path = /services/inbox/default/
use_https = False
use_auth_basic = False
```

```
auth_basic_username = your_username
auth_basic_password = your_password
use_auth_certificate = False
auth_certificate_keyfile = full_path_to_keyfile
auth_certificate_certfile = full_path_to_certfile
include_contact_info = False
contact_name = ...
contact_email = ...

[logstash]
enabled = False
host = localhost
port = 5659
handler = AMQP/TCP/UDP

[misc]
banner = Apache/2.0.48

[surface]
google_meta =
bing_meta =

[sensor]
#sensorid = 501945f7-f4a1-48d5-bb1c-9c183b702909
sensorid = 4
[profiler]
enabled = False
```

Se ha configurado para que no acceda a Logstash porque lo hace a través de Filebeat. También se ha usado MySQL como base de datos y no SQLite. El puerto donde se despliega es el 80.

Por último, para ejecutar Glastopf lo hacemos desde la carpeta "/opt/myhoneypot":

```
$ cd /opt/myhoneypot
$ glastopf-runner
```

A.5 Instalación DionaeaFR

A.5.1 Prerrequisitos

*Antes de iniciar la instalación Dionaea ya debe de estar instalado [62].

Instalamos los paquetes necesarios por apt:

```
$ sudo apt-get install python-pip python-netaddr build-essential python-
dev git
```

Y los paquetes de python por pip:

```
$ sudo pip install Django pygeoip django-pagination django-tables2 django-
compressor django-htmlmin django-filter
```

*Usar las siguientes versiones para no modificar código (se debe a cambios que hicieron en Django y arrastran a todos los plugins):

```
$ sudo pip install Django-compressor==1.4
$ sudo pip install Django==1.6.12
$ sudo pip install Django-filter==0.11.0
$ sudo pip install django-tables2==0.16
```

*Si falla algún paquete probar a actualizar pip:

```
$ sudo pip install --upgrade setuptools
```

Ahora instalamos por repositorio django-tables2-simplefilter:

```
$ cd /opt/
$ wget https://github.com/benjiec/django-tables2-
simplefilter/archive/master.zip -O django-tables2-simplefilter.zip
$ unzip django-tables2-simplefilter.zip
$ mv django-tables2-simplefilter-master/ django-tables2-simplefilter/
$ cd django-tables2-simplefilter/
$ python setup.py install
```

SubnetTree:

```
$ cd /opt/
$ git clone https://github.com/bro/pysubnettree.git
$ cd pysubnettree/
$ python setup.py install
```

Y node.js:

```
$cd /opt/
//$ wget http://nodejs.org/dist/v0.8.16/node-v0.8.16.tar.gz
//$ tar xzvf node-v0.8.16.tar.gz
$ wget http://nodejs.org/dist/latest/node-v7.7.4.tar.gz
$ tar xzvf node-v7.7.4.tar.gz
$ cd node-v7.7.4
$ ./configure
$ make
$ make install
```

Para finalizar:

```
$ npm install -g less
$ npm install -g promise
```

A.5.2 Instalación

Nos bajamos el repositorio:

```
$ cd /opt/
$ git clone https://github.com/rubenespadas/DionaeaFR.git
```

Y algunos plugins de Geolite; GeoIP y GeoLiteCity:

```
$ cd /opt/
$ wget
http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
$ wget
http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.da
t.gz
$ gunzip GeoLiteCity.dat.gz
$ gunzip GeoIP.dat.gz
$ mv GeoIP.dat DionaeaFR/DionaeaFR/static
$ mv GeoLiteCity.dat DionaeaFR/DionaeaFR/static
```

*También hay que cambiar el "settings.py.dist" a "settings.py" y asegurarse de mirar (dentro del archivo) que la base de datos que utiliza es "'/opt/dionaea/var/dionaea/dionaea.sqlite'".

Y finalmente iniciamos el servidor DionaeaFR:

```
$ mkdir /var/run/dionaeafr
$ python manage.py collectstatic
$ python manage.py runserver 0.0.0.0:8000
```

*Si falta un import de "setting_changed" cambiar en "/usr/local/lib/python2.7/dist-packages/django_filters/conf.py"

```
"from django.core.signals import setting_changed"
```
a
```
"from django.test.signals import setting_changed"
```

*Si falla por el tag "nospacesless" renombrarlo a "spaceless"

A.6 Instalación HoneyDrive

A.6.1 Prerrequisitos

Tener espacio para una máquina virtual nueva.
Descargar el archivo ova de su página web (es directamente una máquina virtual) [https://sourceforge.net/projects/honeydrive/].

A.6.2 Instalación

Una vez descargado, desde VMware o VirtualBox, importar la máquina virtual con extensión ".ova" y configurar la red para que esté en modo promiscuo y con adaptador puente al conector que usemos nosotros normalmente para la red.

Los principales honeypots que contiene son los mismos que los que se han explicado individualmente en apartados anteriores.

A.7 Instalación ELK Stack (ElasticSearch, Logstash y Kibana)

La arquitectura ELK Stack, está enfocada a recoger registros o *logs* y mostrarlos de una forma cómoda para analizarlos. Debido a que los datos que generan los honeypots con las caapturas de ataques son *logs*, esta herramienta nos resulta de gran utilidad.

A.7.1 Prerrequisitos

Tener instalado Java 8, para ello añadimos el repositorio de Oracle Java a nuestra lista de del comando "apt":

```
$ sudo add-apt-repository -y ppa:webupd8team/java
```

Ejecutamos update:

```
$ sudo apt-get update
```

E instalamos el paquete:

```
$ sudo apt-get -y install oracle-java8-installer
```

A.7.2 Instalación

El ELK Stack se compone de 4 módulos principales [63], como se puede ver en la Figura 4-3:

- **Logstash**: procesa los logs que llegan al servidor.
- **Elasticsearch**: almacena los logs que ha procesado Logstash.
- **Kibana**: es una interfaz web que representa los logs almacenados en Elasticsearch.
- **Filebeat**: se instala en los sensores y envía los logs que se generan al servidor donde se encuentre Elasticsearch.

El orden de instalación en el que se realizará el despliegue del sistema es: Elasticsearch, Kibana, Logstash y finalmente Filebeat.

Instalación Elasticsearch:

Importamos a la lista de apt el repositorio de Elasticsearch:

```
$ wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-
key add –

$ echo "deb http://packages.elastic.co/elasticsearch/2.x/debian stable
main" | sudo tee -a /etc/apt/sources.list.d/elasticsearch-2.x.list
```

Hacemos update:

```
$ sudo apt-get update
```

E instalamos el paquete:

```
$ sudo apt-get -y install elasticsearch
```

Ahora configuramos Elasticsearch para que no se pueda acceder externamente:

```
$ sudo gedit /etc/elasticsearch/elasticsearch.yml
```

Y lo configuramos para que únicamente se pueda acceder desde localhost:

```
network.host: localhost
```

Ahora lo ejecutamos y creamos un daemon:

```
$ sudo systemctl restart elasticsearch
$ sudo systemctl daemon-reload
$ sudo systemctl enable elasticsearch
```

Instalación Kibana:

Importamos a la lista de apt el repositorio de Kibana:

```
$ echo "deb http://packages.elastic.co/kibana/4.5/debian stable main" |
sudo tee -a /etc/apt/sources.list
```

Hacemos update:

```
$ sudo apt-get update
```

E instalamos el paquete:

```
$ sudo apt-get -y install kibana
```

Una vez instalado configuramos Kibana para que se pueda acceder a ella externamente. Como disponemos de un dominio usaremos dicho dominio para publicar el servidor. En el siguiente archivo de configuración:

```
$ sudo gedit /opt/kibana/config/kibana.yml
```

Cambiamos el siguiente parámetro:

```
server.host: "rmartinez.ii.uam.es"
```

Y por último creamos el daemon:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable kibana
$ sudo systemctl start kibana
```

Instalación Logstash:

Importamos a la lista de apt el repositorio de Logstash:

```
$ echo "deb http://packages.elastic.co/logstash/2.3/debian stable main" |
sudo tee -a /etc/apt/sources.list
```

Hacemos update:

```
$ sudo apt-get update
```

E instalamos el paquete:

```
$ sudo apt-get -y install logstash
```

A continuación, generamos los certificados SSL para comunicar los sensores que ejecutarán Filebeat con nuestro servidor Logstash:

```
$ sudo mkdir -p /etc/pki/tls/certs
$ sudo mkdir /etc/pki/tls/private

$ cd /etc/pki/tls
$ sudo openssl req -subj '/CN=ELK_server_fqdn/' -x509 -days 3650 -batch -
nodes -newkey rsa:2048 -keyout private/logstash-forwarder.key -out
certs/logstash-forwarder.crt
```

Ahora configuramos Logstash para que use los certificados. Aunque son auto-firmados, no necesitamos ninguna identidad externa que verifique la identidad de nuestro servidor, pues somos nosotros los que principalmente accedemos a él:

```
$ sudo nano /etc/logstash/conf.d/02-beats-input.conf

input {
  beats {
    port => 5044
    ssl => true
    ssl_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"
    ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
  }
}
```

Y creamos un archive de prueba para filtrar syslogs:

```
$ sudo nano /etc/logstash/conf.d/10-syslog-filter.conf

filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}
%{SYSLOGHOST:syslog_hostname}
%{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?:
%{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    syslog_pri { }
    date {
      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss"
]
    }
  }
```

```
}
```

Y ahora el output que recibe Elasticsearch

```
$ sudo nano /etc/logstash/conf.d/30-elasticsearch-output.conf

output {
  elasticsearch {
    hosts => ["localhost:9200"]
    sniffing => true
    manage_template => false
    index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
    document_type => "%{[@metadata][type]}"
  }
}
```

Para finalizar comprobamos que la configuración se ha realizado correctamente:

```
$ sudo /opt/logstash/bin/logstash --configtest -f /etc/logstash/conf.d/ --
--path.settings=/etc/logstash
```

Y creamos su daemon:

```
$ sudo systemctl restart logstash
$ sudo systemctl enable logstash
```

Instalación Filebeat:

Ahora vamos a instalar una instancia de Filebeat en cada sensor. Las siguientes instrucciones son ejecutadas desde el lado del sensor. Primero hay que pasarle el certificado SSL que hemos generado anteriormente y copiarlo en la siguiente carpeta:

```
$ sudo mkdir -p /etc/pki/tls/certs
$ sudo cp /tmp/logstash-forwarder.crt /etc/pki/tls/certs/
```

Importamos a la lista de apt el repositorio de Filebeat:

```
$ echo "deb https://packages.elastic.co/beats/apt stable main" |  sudo tee
-a /etc/apt/sources.list.d/beats.list

$ wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-
key add -
```

Hacemos update:

```
$ sudo apt-get update
```

E instalamos el paquete:

```
$ sudo apt-get -y install filebeat
```

La configuración de Filebeat dependerá del tipo de sensor en el que esté instalado. En el apéndice de configuraciones se detallan las configuraciones para cada sensor.

*Para debug de Filebeat:

```
$ /usr/share/filebeat/bin/filebeat -c /etc/filebeat/filebeat.yml -e -d "*"
```

B. Manual del programador

B.1 Configuración Filebeat

La configuración de Filebeat se realiza a través del archivo "`/etc/filebeat/filebeat.yml`". A continuación, se muestran los parámetros que debe contener para enviar los archivos al servidor que contiene la base de datos.

B.1.1 Configuración Filebeat Glastopf-Cowrie

La configuración usada ha sido la siguiente:

```
filebeat.prospectors:

- input_type: log

    # Path to Cowrie's logs
     - /home/cowrie/cowrie/log/cowrie.json*
  document_type: cowrie
  max_bytes: 2097152000
  harvester_buffer_size: 2097152

- input_type: log
  paths:
    - /home/logs/glastopf/glastopf.log
  document_type: glastopf
  max_bytes: 2097152000
  harvester_buffer_size: 2097152

#Name of the beat
name: GlasCowrie1

output.logstash:
  # The Logstash hosts
  hosts: ["direccionservidor.uam.es:5044"]
```

Como se observa, los campos dentro de prospectors indican los parámetros para recoger los logs, como la carpeta de origen, el tipo de documento o el tamaño máximo disponible para leer el log.

El parámetro "name" permite identificar posteriormente en Elasticsearch y Kibana el sensor por su nombre (es recomendable que sea único para cada sensor).
El output elegido en nuestro caso es Logstash, aunque podría ser Elasticsearch directamente pero no se aplicarían las modificaciones ni filtros que están definidos en Logstash antes de agregarlos a la base de datos.

B.1.2 Configuración Filebeat Dionaea

Al igual que en Glastopf y Cowrie en la máquina donde se encuentra Dionaea configuramos Filebeat con el siguiente formato:

```
filebeat.prospectors:

- input_type: log

paths:
  - /home/logs/dionaea/dionaea.log
document_type: dionaea
max_bytes: 2097152000
harvester_buffer_size: 2097152

#Name of the beat
name: Dionaea1

output.logstash:
  # The Logstash hosts
  hosts: ["direccionservidor.uam.es:5044"]
```

B.2 Configuración Logstash

En Logstash hay que configurar los filtros para cada tipo de honeypot. Como ya hemos indicado en Filebeat el tipo de documento que es, en Logstash filtraremos por dicho tipo e indicaremos a través de patrones el contenido de cada campo.

B.2.1 Configuración Logstash Glastopf

Para configurar el puerto donde Logstash escucha a Filebeat hay que añadir un archivo de configuración en la carpeta "/etc/logstash/conf.d", en nuestro caso le hemos llamado "honeys.conf".

Dentro de ese fichero colocaremos la siguiente configuración:

```
#Indicar el puerto donde se escucha a Filebeat
input {
  beats {
      port => 5044    # Pick an available port to listen on
      #congestion_threshold => 10000
    }
}

#Filtros que se aplican
filter {
    if [type] == "glastopf" {
        grok {
          match => [ "message", "%{YEAR:year}-%{MONTHNUM2:month}-
%{MONTHDAY:day} %{TIME:time} : %{IP:src_ip} \t%{DATA:src_port} \t %{IP:dst_ip}
\t %{DATA:dst_port} \t %{DATA:protocol} \t %{DATA:request_url} \t
%{DATA:pattern} \t %{DATA:filename} \t %{DATA:request_method} \t
'%{DATA:request_raw}' " ]

        }
        mutate {
          strip => [ "src_ip", "dst_ip", "protocol", "src_port" , "dst_port",
"pattern", "filename", "request_url" ]
```

```
        }

        mutate {
          add_field => [ "timestamp", "%{year}-%{month}-%{day}:%{time}" ]
        }
        date {
          match => [ "timestamp" , "yyyy-MM-dd:HH:mm:ss" ]
        }

        mutate {
          add_field => [ "basetype", "honeypot" ]
        }

        if [src_ip]  {

            dns {
                reverse => [ "src_host", "src_ip" ]
                action => "append"
            }

            geoip {
                source => "src_ip"  # With the src_ip field
                target => "geoip"   # Add the geoip one
                # Using the database we previously saved
                database => "/opt/logstash/vendor/geoip/GeoLite2-City.mmdb"
                add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}"
]
                add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}"
]
            }

            mutate {
                convert => [ "[geoip][coordinates]", "float" ]
            }
        }
    }
}

#Indicar donde se almacenan los datos filtrados: como Logstash está en el mismo
servidor que Elasticsearch podemos indicarle que vaya a localhost
output {
    if [type] == "glastopf" {
        # Output to elasticsearch
        elasticsearch {
           hosts => ["localhost:9200"]  # Provided elasticsearch is listening
on that host:port
           sniffing => true
           manage_template => false
           index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
           document_type => "%{[@metadata][type]}"
        }
        # For debugging
        stdout {
            codec => rubydebug
        }
    }
}
```

Como se puede observar hay 3 grandes campos: input, filter y output.

- Input: indica el puerto donde hay que escuchar y el tipo, en este caso Filebeat.
- Filter: recibimos el archivo de texto y a través de los patrones que declaramos sacamos los campos para almacenar en nuestra base de datos (en este caso Elasticsearch).
- Output: indica a donde se redirigen los datos filtrados, en nuestro caso se redirigen a Elasticsearch por el puerto 9200.

B.2.2 Configuración Logstash Cowrie

Partiendo de la configuración de Glastopf añadimos un nuevo filtro para Cowrie y un nuevo output (por si en un futuro es necesario redirigirlo a un destino distinto).

```
filter{
#Aquí está incluido el de Glastopf
if [type] == "cowrie" {

        json {
            source => message
        }

        mutate {
            rename => { "system" => "system_cowrie" }
        }


        date {
            match => [ "timestamp", "ISO8601" ]
        }

        if [src_ip]  {

            dns {
                reverse => [ "src_host", "src_ip" ]
                action => "append"
            }

            geoip {
                source => "src_ip"  # With the src_ip field
                target => "geoip"   # Add the geoip one
                # Using the database we previously saved
                database => "/opt/logstash/vendor/geoip/GeoLite2-City.mmdb"
                add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}"
]
                add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}"
]
            }

            mutate {
                convert => [ "[geoip][coordinates]", "float" ]
            }
        }
    }
}
```

Y el nuevo output también va incluido, pero es similar al de Glastopf y no merece la pena repetirlo.

B.2.3 Configuración Logstash Dionaea

Al igual que Glastopf y Cowrie va en el mismo archivo de configuración con el siguiente filtro:

```
if [type] == "dionaea" {
        grok {
           match => [ "message", "%{YEAR:year}-%{MONTHNUM2:month}-
%{MONTHDAY:day} %{TIME:time} : %{DATA:connection_type} \t
%{DATA:connection_protocol}\t%{DATA:protocol}\t %{IP:src_ip} \t%{DATA:src_port}
\t %{IP:dst_ip} \t %{DATA:dstport} \t %{DATA:hostname}" ]
        }
        mutate {
          strip => [ "connection_protocol", "connection_type", "protocol",
"src_port" , "dst_port", "hostname" ]
        }

        mutate {
          add_field => [ "timestamp", "%{year}-%{month}-%{day}:%{time}" ]
        }
        date {
          match => [ "timestamp" , "yyyy-MM-dd:HH:mm:ss" ]
        }

        if [src_ip]  {
            geoip {
                source => "src_ip"  # With the src_ip field
                target => "geoip"   # Add the geoip one
                # Using the database we previously saved
                database => "/opt/logstash/vendor/geoip/GeoLite2-City.mmdb"
                add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}"
]
                add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}"
]
            }

            # Get the ASN code as well
            #geoip {
            #    source => "src_ip"
            #    database => "/opt/logstash/vendor/geoip/GeoLite2-ASN.mmdb"
            #}

            mutate {
                convert => [ "[geoip][coordinates]", "float" ]
            }
        }
    }
```

B.3 Creación de scripts para generar logs en texto

Estos scripts son similares a hacer un “dump” de una base de datos: buscan en una o varias tablas y lo escriben en un documento de texto. Dichos scripts se ejecutan automáticamente

en las máquinas virtuales donde están los honeypots (y sus bases de datos) y van extrayendo las nuevas filas automáticamente.

Para que se ejecuten los scripts se ha usado crontab abriendo su archivo de configuración:

```
$ crontab -e
```

Y agregando los scripts para que se ejecuten cada media hora:

```
*/30 * * * * /home/glastopf_log.py
*/30 * * * * /home/dionaea_log.py
```

Estos scripts crean los documentos de texto o "logs" que Filebeat envía. En el caso de Cowrie no es necesario pues directamente crea unos archivos ".json" que Logstash es capaz de interpretar.

B.3.1 Script para Dionaea

Para Dionaea hay que acceder a la base de datos SQLite que tiene y extraer las filas de la tabla "connections". Para saber cuál fue la última fila extraída se escribe un fichero de texto con el último "id". Gracias al usuario "cudeso" [64] disponemos de una plantilla muy simple en Python sobre cómo sacar los datos.

El funcionamiento del script es muy sencillo:

- La variable LAST_CONNECTION_FILE: contiene la dirección de un archivo de texto que indica el número o identificador del último registro que se ha extraído.
- SQLITE_DB: indica la ubicación de la base de datos creada por Dionaea. Se accede a la tabla "connections" y se extraen los registros para imprimirlos en un archivo de texto ".log" (podría ser ".txt", pero así sabemos que es un *log*).
- LOGFILE: en este archivo de texto es donde se almacenan los registros de la tabla "connections" que enviará Filebeat al servidor.
- IGNORE_SRC: se utiliza para obviar aquellas conexiones que no van únicamente a Dionaea.

```
import os
import sys
import datetime
import sqlite3

SQLITE_DB = "/opt/dionaea/var/dionaea/dionaea.sqlite"
LAST_CONNECTION_FILE = "/home/dionaea_last_log.txt"
LOGFILE="/home/logs/dionaea/dionaea.log"
IGNORE_SRC=[ "127.0.0.1" ]

connection_start = 0
connection_id = 0

if __name__ == "__main__":

    if os.path.isfile(SQLITE_DB):

        if os.path.isfile(LAST_CONNECTION_FILE):
```

```
            f = open(LAST_CONNECTION_FILE, 'r')
            f_content = f.read()
            f.close()
            if f_content and int(f_content) > 0:
                connection_start = int(f_content)

        conn = sqlite3.connect(SQLITE_DB)
        c = conn.cursor()

        if LOGFILE:
            f_log = open(LOGFILE, 'a')
        for row in c.execute("SELECT * FROM connections WHERE connection > %s
ORDER BY connection ASC" % connection_start):
            timestamp = datetime.datetime.fromtimestamp(row[4]).strftime('%Y-
%m-%d %H:%M:%S')
            connection_type = row[1]
            protocol = row[2]
            connection_protocol = row[3]
            dst_ip = row[7]
            dst_port = row[8]
            src_ip = row[9]
            src_port = row[11]
            hostname = row[10]
            connection_id = row[0]
            if src_ip in IGNORE_SRC:
                continue
            if connection_protocol == "p0fconnection":
                continue
            if LOGFILE:
                f_log.write("%s : %-10s \t %-10s \t %s \t %s \t %s \t %s \t %s
\t %s\n" % (timestamp, connection_type, connection_protocol, protocol, src_ip,
src_port, dst_ip, dst_port, hostname))
            else:
                print "%s : %-10s \t %-10s \t %s \t %s \t %s \t %s \t %s \t %s
" % (timestamp, connection_type, connection_protocol, protocol, src_ip,
src_port, dst_ip, dst_port, hostname)
        conn.close()
        if LOGFILE:
            f_log.close()

        if not(connection_id and connection_id > 0):
            connection_id = connection_start
        f = open(LAST_CONNECTION_FILE, 'w')
        f.write(str(connection_id))
        f.close()
    else:
        print "Sqlite DB not found : %s " % SQLITE_DB
```

B.3.2 Script para Glastopf

El código para Glastopf es muy parecido al de Dionaea, pero cambiando la base de datos:

```
import os
import sys
```

```
import datetime
import sqlite3
import MySQLdb

DSTIP="150.244.57.39" #La IP de Glastopf
DSTPORT="80"
PROTOCOL="tcp"

LOGFILE= "/home/glastopf_last_log.txt"
LAST_CONNECTION_FILE = "/home/logs/glastopf/glastopf.log"


SQLITE_DB="/opt/myhoneypot/db/glastopf.db"

DB_USER="glaspot"
DB_PASS="glaspot"
DB_DB="glaspot"
DB_HOST="localhost"

connection_start = 0
connection_id = 0

if __name__ == "__main__":

    if os.path.isfile(LAST_CONNECTION_FILE):
        f = open(LAST_CONNECTION_FILE, 'r')
        f_content = f.read()
        f.close()
        if f_content and int(f_content) > 0:
            connection_start = int(f_content)

    #if SQLITE_DB:
    #    db = sqlite3.connect(SQLITE_DB)
    #else:
    #    db = MySQLdb.connect(host=DB_HOST, user=DB_USER, passwd=DB_PASS,
db=DB_DB)

    db = MySQLdb.connect(host=DB_HOST, user=DB_USER, passwd=DB_PASS, db=DB_DB)
    cur = db.cursor()

    if LOGFILE:
        f_log = open(LOGFILE, 'a')
    cur.execute("SELECT * FROM events WHERE id > %(connect)s AND id < %(connect)s
+ 200000 ORDER BY id ASC" % {'connect': connection_start})
    for row in cur.fetchall() :
        connection_id = row[0]
        timestamp = row[1]
        source = row[2].split(':')
        srcip = source[0]
        srcport = source[1]
        request_url = row[3]
        request_raw = row[4].replace('\n', '|').replace('\r', '')
        request_type = request_raw[0:4].strip()
        if not(request_type == "POST" or request_type == "GET" or request_type ==
"HEAD"):
            request_type = "Unknown"
        pattern  = row[5]
        filename = row[6]
```

```
        if LOGFILE:
            f_log.write("%s : %s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \t %s
\t '%s' \n" % (timestamp, srcip, srcport, DSTIP, DSTPORT, PROTOCOL, request_url,
pattern, filename, request_type, request_raw))
        else:
            print "%s : %s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \t
'%s' \n" % (timestamp, srcip, srcport, DSTIP, DSTPORT, PROTOCOL, request_url,
pattern, filename, request_type, request_raw)
    db.close()
    if LOGFILE:
        f_log.close()

    if not(connection_id and connection_id > 0):
        connection_id = connection_start
    f = open(LAST_CONNECTION_FILE, 'w')
    f.write(str(connection_id))
    f.close()
```

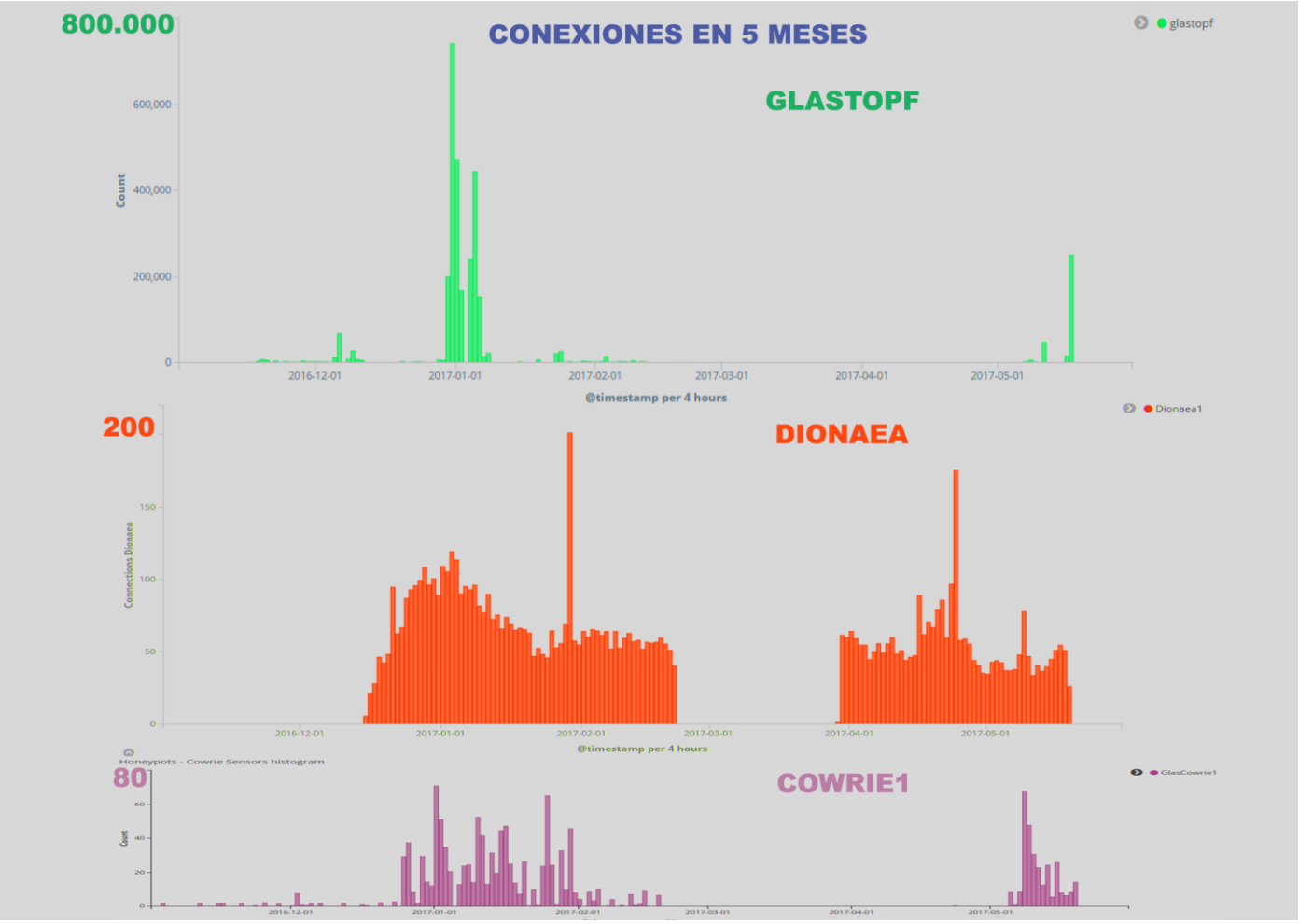
Se ha puesto un límite de 200.000 entradas para que no se sobrecargue Filebeat.

C. Gráficos de los resultados obtenidos

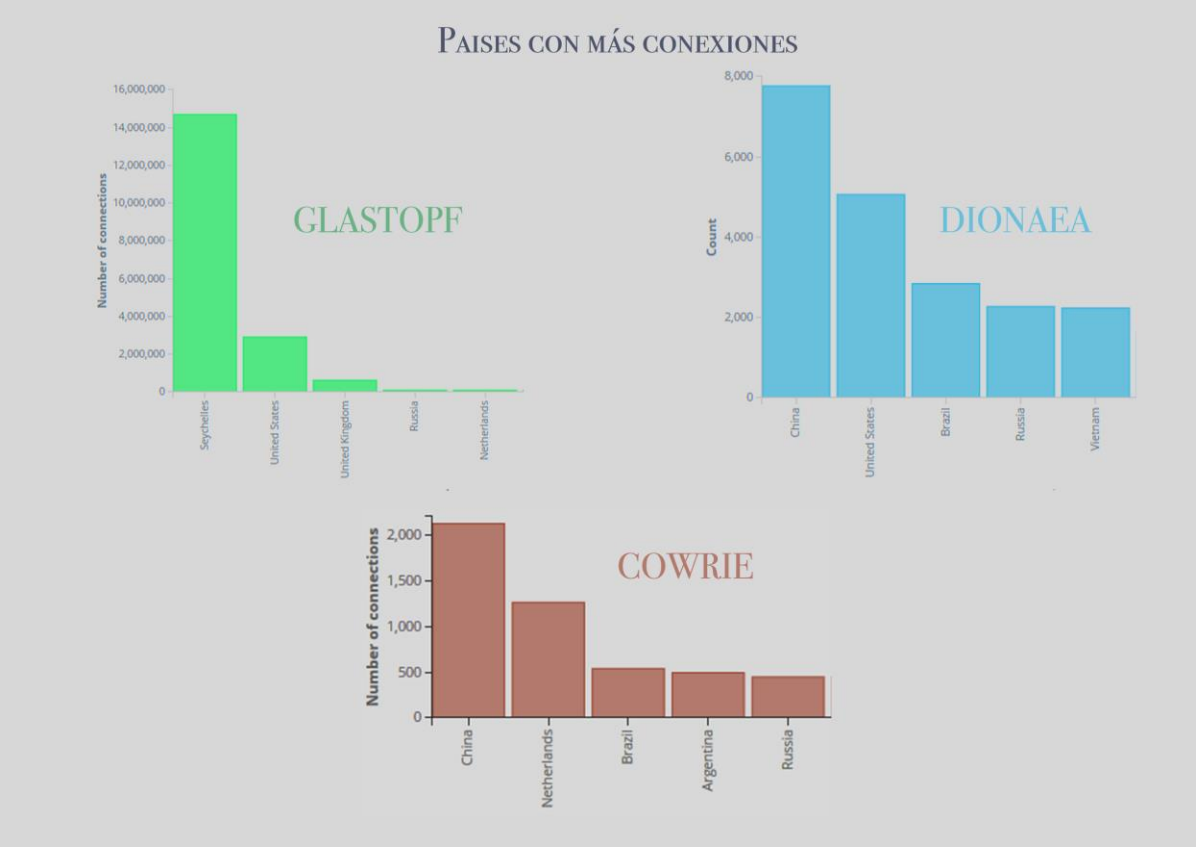
Los siguientes gráficos se han obtenido de Kibana. Kibana permite exponer los datos en distintas formas de gráficos: tablas, histogramas, diagramas circulares, etc. Dichos gráficos hay que configurarlos previamente indicando el tipo de búsqueda, los filtros y el formato. Además, se pueden guardar para posteriores reutilizaciones y se pueden agrupar en paneles llamados "dashboards" en Kibana.

Se han configurado tres "dashboards" que representan la comparación de los 3 tipos de honeypots: un "dashboard" para comparar Dionaea, otro para Cowrie y otro para Glastopf.

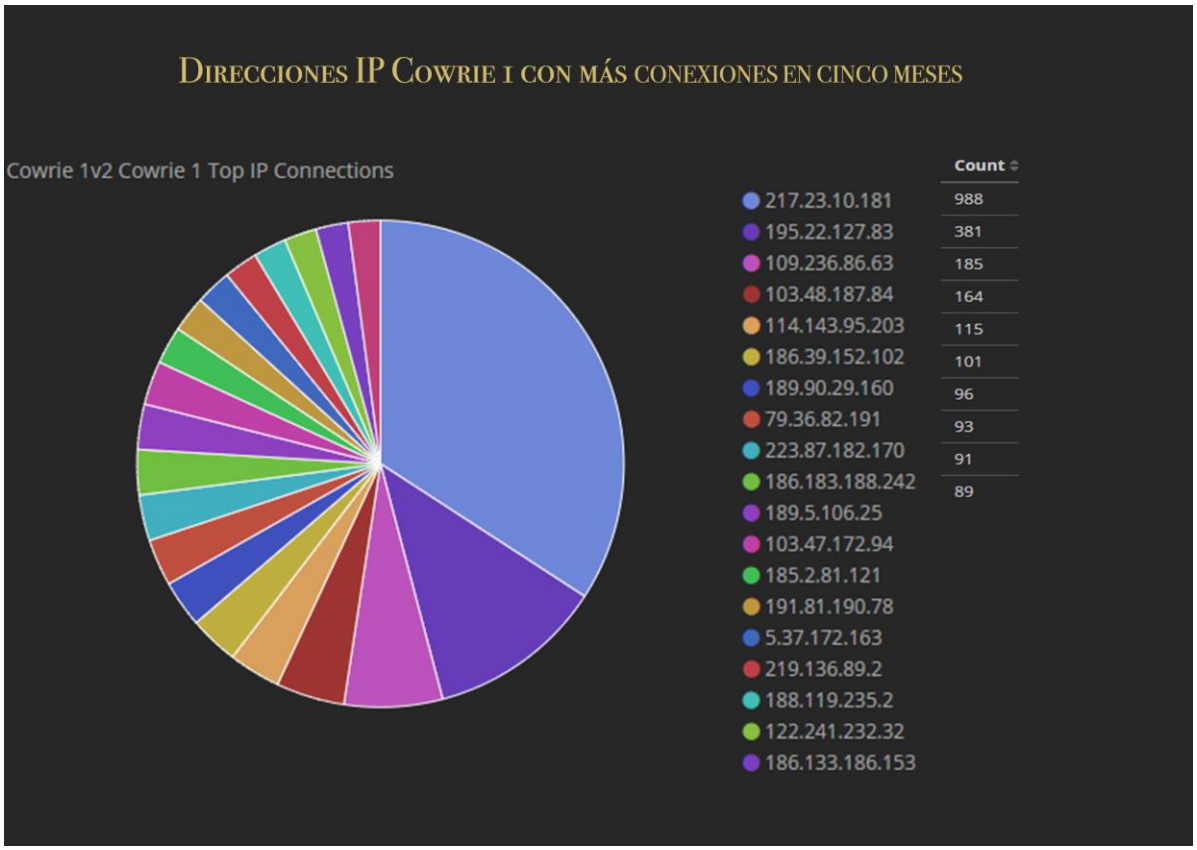
C.1 Resultados Etapa Previa



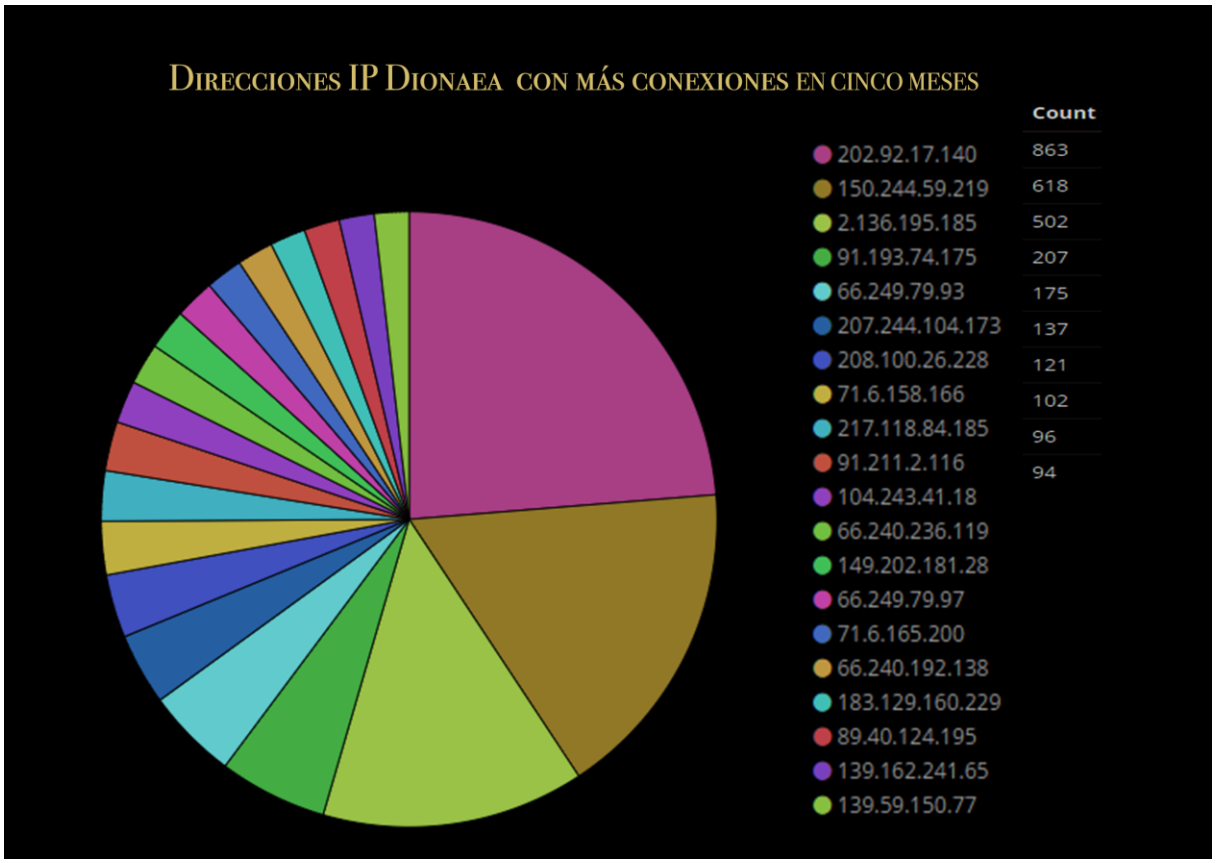
Gráficos Resultados 1- Histogramas Glastopf, Dionaea y Cowrie (E. Previa)



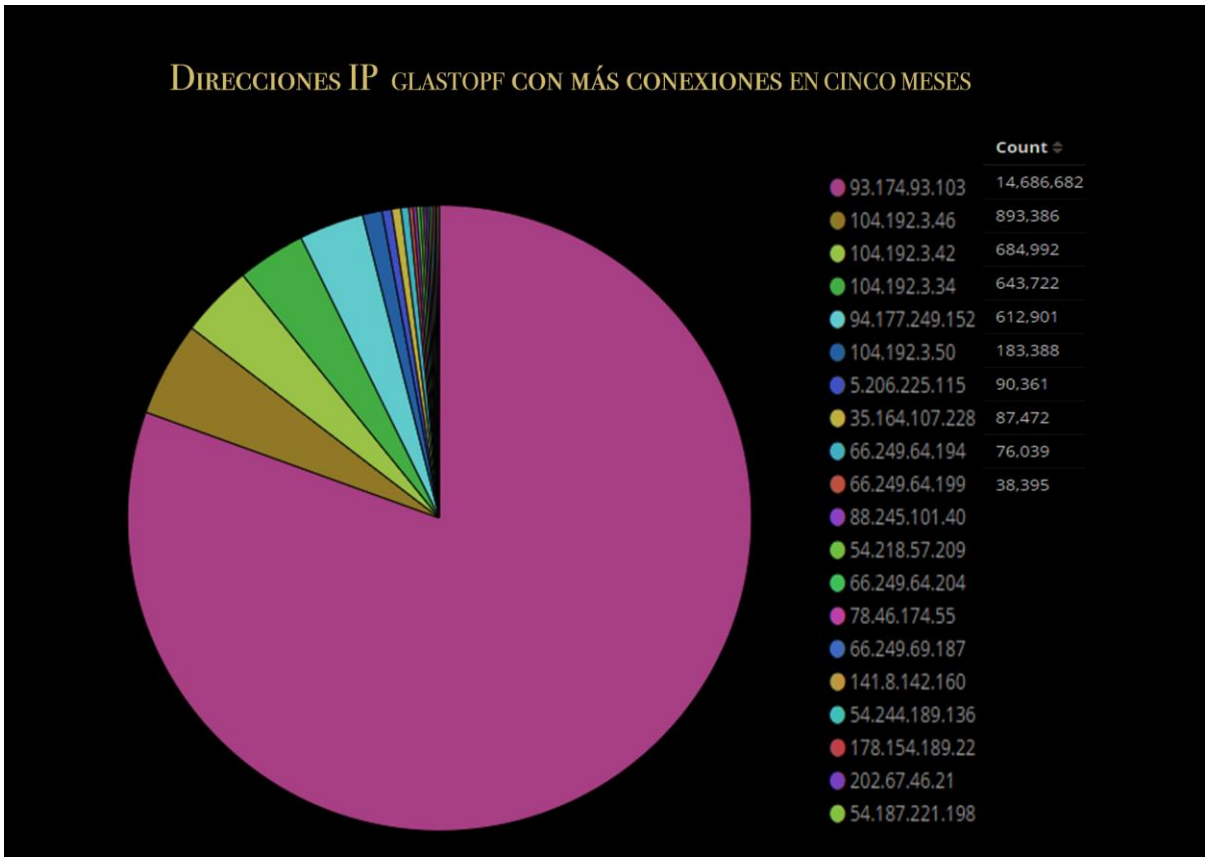
Gráficos Resultados 2- Top 5 Países por honeypot (E. Previa)



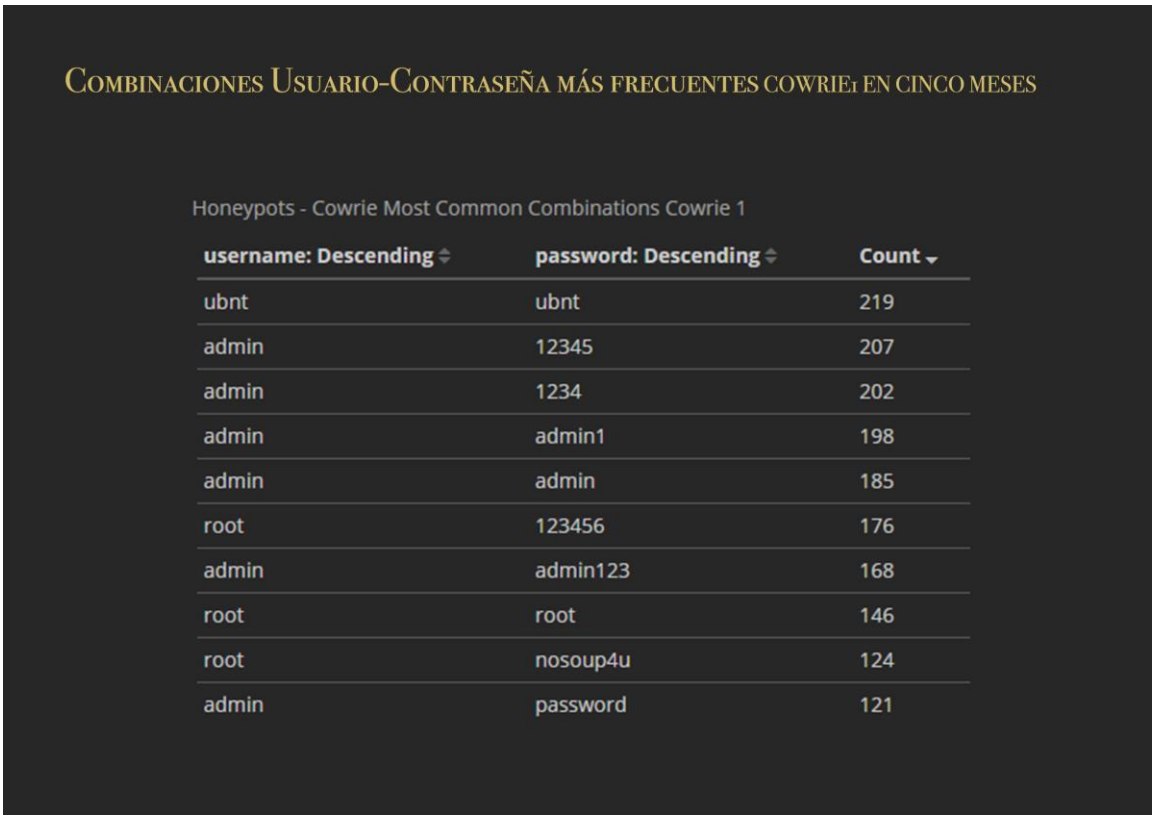
Gráficos Resultados 3- Top IPs Cowrie (E. Previa)



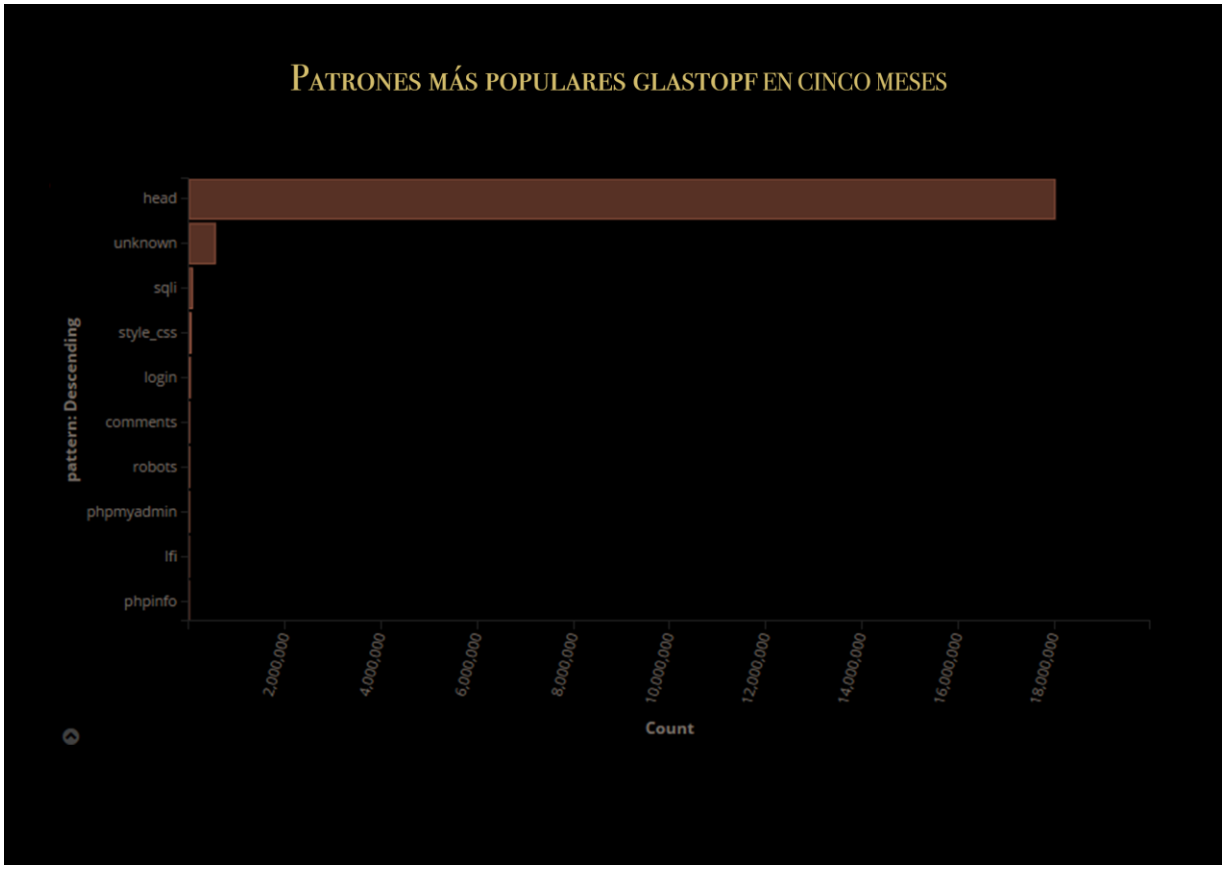
Gráficos Resultados 4- Top IPs Dionaea (E. Previa)



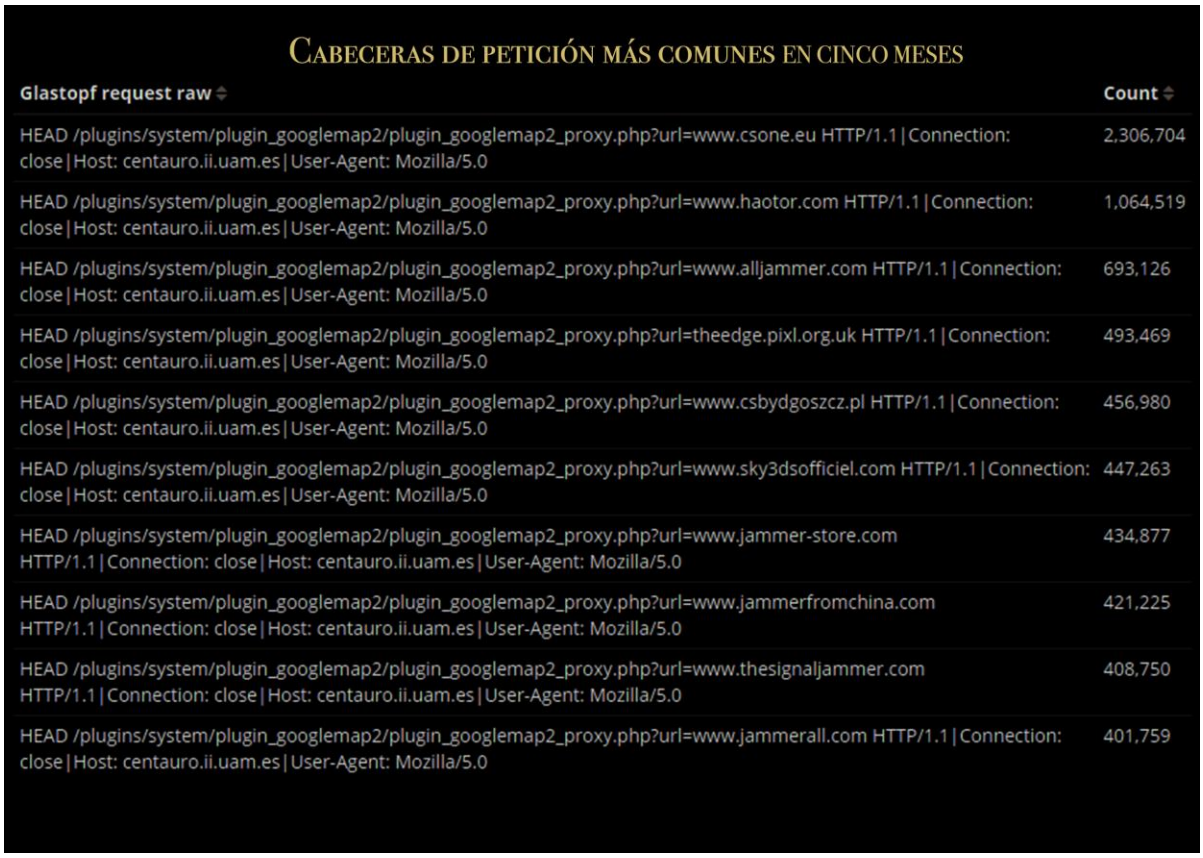
Gráficos Resultados 5- Top IPs Glastopf (E. Previa)



Gráficos Resultados 6- Top Combinaciones Usuario-Contraseña (E. Previa)



Gráficos Resultados 7- Top Patrones Glastopf (E. Previa)



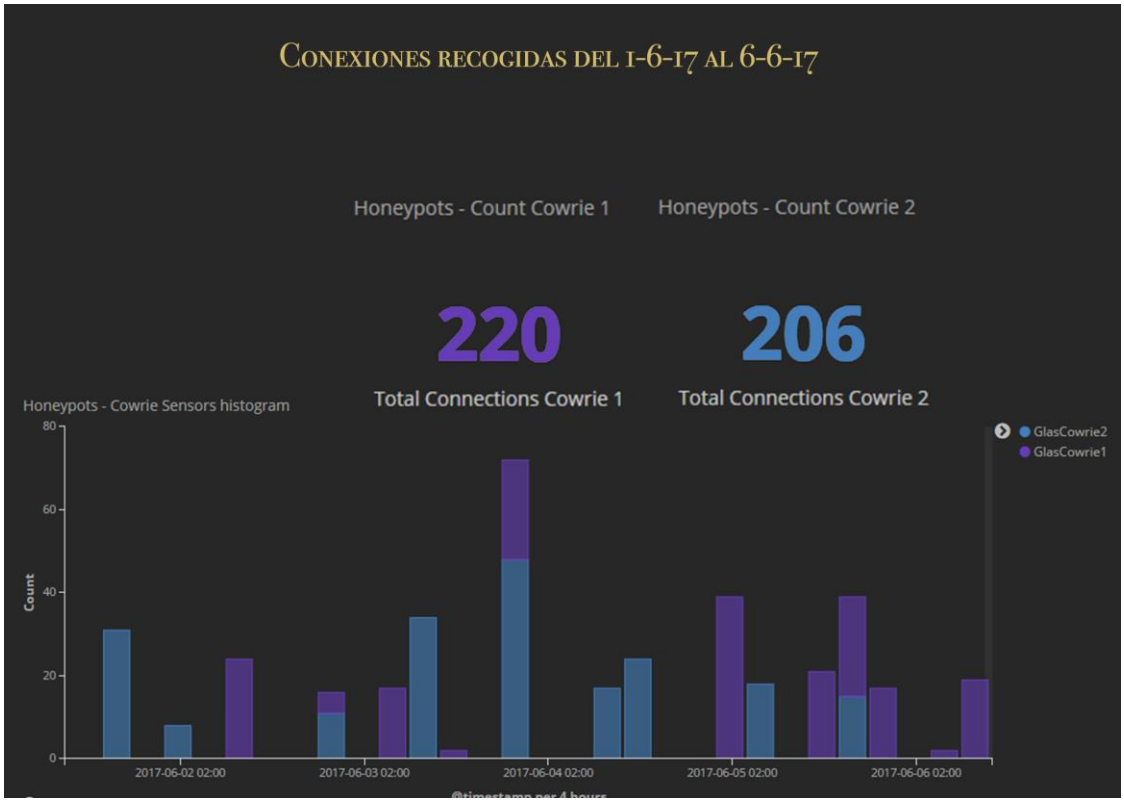
CABECERAS DE PETICIÓN MÁS COMUNES EN CINCO MESES

Glastopf request raw	Count
HEAD /plugins/system/plugin_googlemap2/plugin_googlemap2_proxy.php?url=www.csone.eu HTTP/1.1|Connection: close|Host: centauro.ii.uam.es|User-Agent: Mozilla/5.0	2,306,704
HEAD /plugins/system/plugin_googlemap2/plugin_googlemap2_proxy.php?url=www.haotor.com HTTP/1.1|Connection: close|Host: centauro.ii.uam.es|User-Agent: Mozilla/5.0	1,064,519
HEAD /plugins/system/plugin_googlemap2/plugin_googlemap2_proxy.php?url=www.alljammer.com HTTP/1.1|Connection: close|Host: centauro.ii.uam.es|User-Agent: Mozilla/5.0	693,126
HEAD /plugins/system/plugin_googlemap2/plugin_googlemap2_proxy.php?url=theedge.pixl.org.uk HTTP/1.1|Connection: close|Host: centauro.ii.uam.es|User-Agent: Mozilla/5.0	493,469
HEAD /plugins/system/plugin_googlemap2/plugin_googlemap2_proxy.php?url=www.csbydgoszcz.pl HTTP/1.1|Connection: close|Host: centauro.ii.uam.es|User-Agent: Mozilla/5.0	456,980
HEAD /plugins/system/plugin_googlemap2/plugin_googlemap2_proxy.php?url=www.sky3dsofficiel.com HTTP/1.1|Connection: close|Host: centauro.ii.uam.es|User-Agent: Mozilla/5.0	447,263
HEAD /plugins/system/plugin_googlemap2/plugin_googlemap2_proxy.php?url=www.jammer-store.com HTTP/1.1|Connection: close|Host: centauro.ii.uam.es|User-Agent: Mozilla/5.0	434,877
HEAD /plugins/system/plugin_googlemap2/plugin_googlemap2_proxy.php?url=www.jammerfromchina.com HTTP/1.1|Connection: close|Host: centauro.ii.uam.es|User-Agent: Mozilla/5.0	421,225
HEAD /plugins/system/plugin_googlemap2/plugin_googlemap2_proxy.php?url=www.thesignaljammer.com HTTP/1.1|Connection: close|Host: centauro.ii.uam.es|User-Agent: Mozilla/5.0	408,750
HEAD /plugins/system/plugin_googlemap2/plugin_googlemap2_proxy.php?url=www.jammerall.com HTTP/1.1|Connection: close|Host: centauro.ii.uam.es|User-Agent: Mozilla/5.0	401,759

Gráficos Resultados 8- Cabeceras de peticiones Glastopf

C.2 Resultados análisis – ambos sistemas detrás del Firewall

C.2.1 Resultados Cowrie

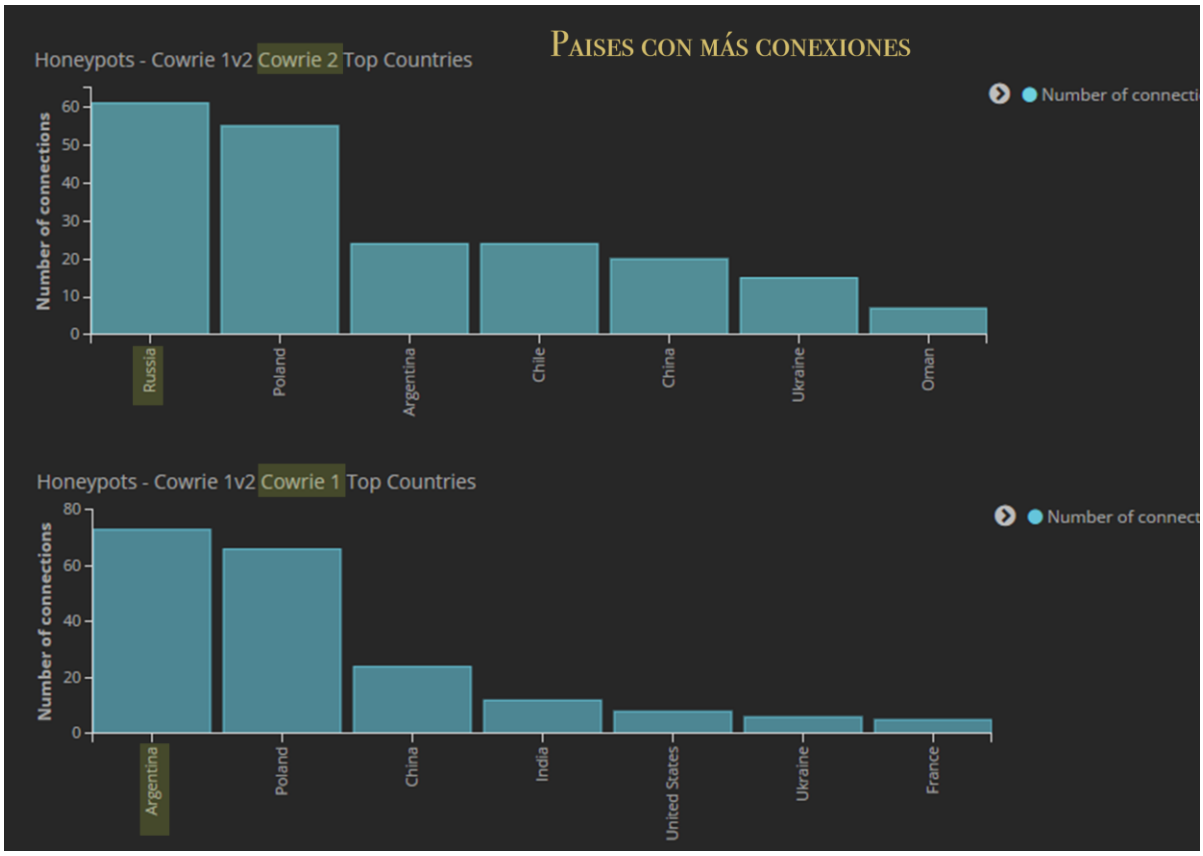


Gráficos Resultados 9- Conexiones Cowrie 1 vs 2

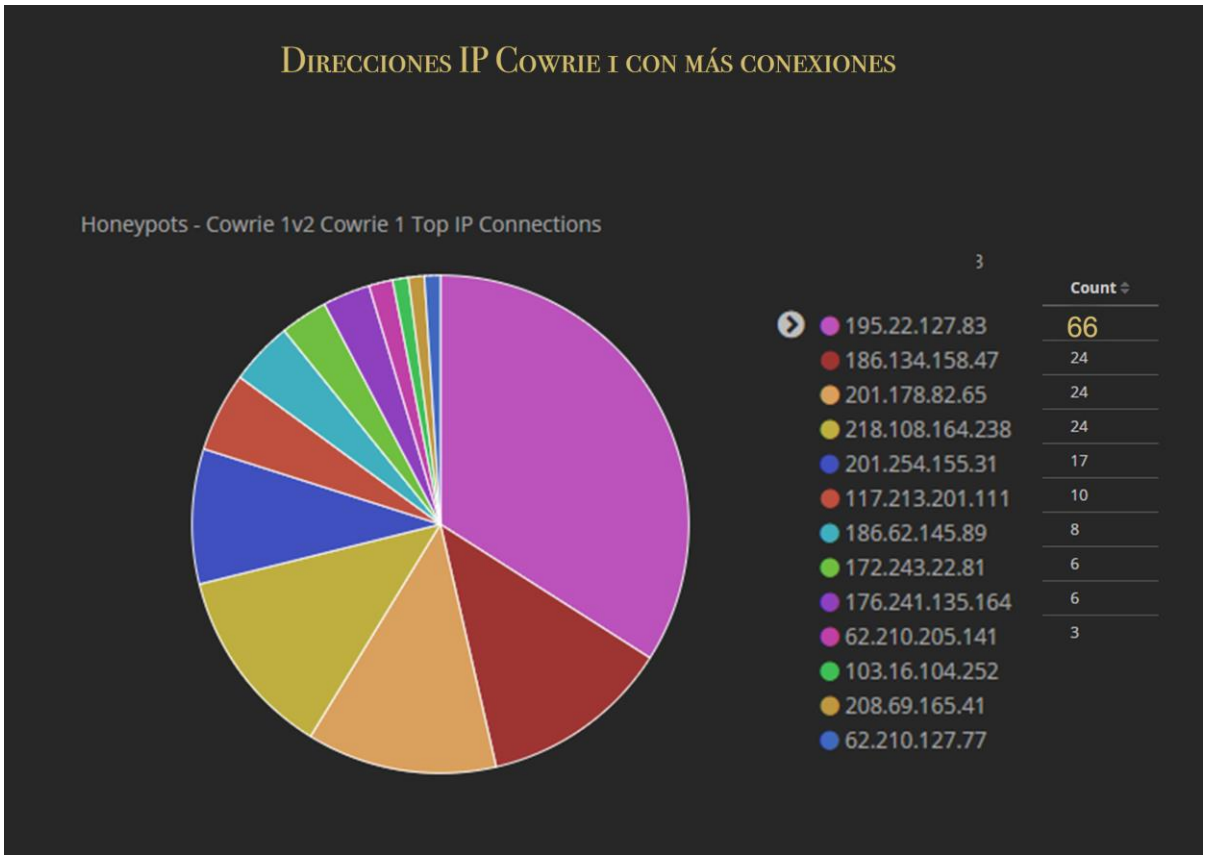
username: Descending	password: Descending	Count
admin	1234	12
admin	admin1234	11
admin	admin1	10
admin	12345	10
admin	password	9
root	root	8
admin	admin123	8
root	123456	4
root	000000	3
admin	admin	3

username: Descending	password: Descending	Count
admin	admin123	11
admin	admin1234	7
admin	admin1	7
root	root	7
root	password	6
root	dreambox	6
admin	password	5
admin	admin	5
admin	1234	5
root	xmhdipc	5

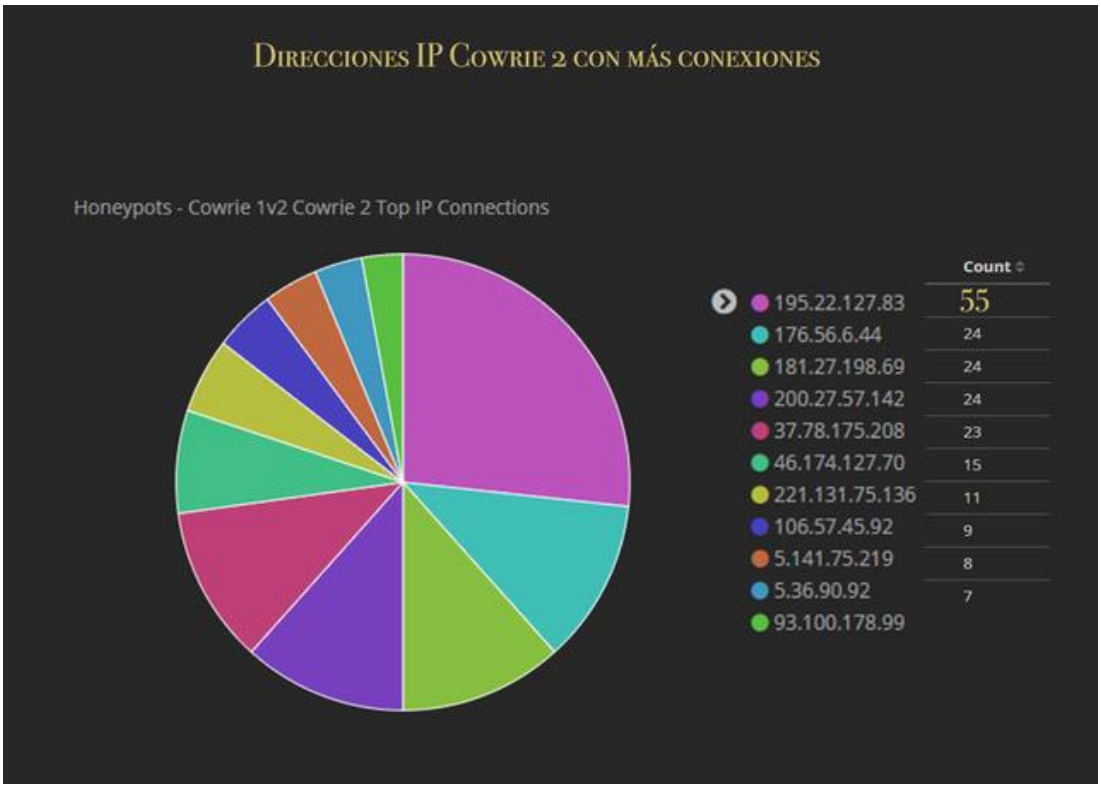
Gráficos Resultados 10- Combinaciones Usuario-Contraseña Cowrie 1 y 2



Gráficos Resultados 11- Top Países Cowrie 1 y 2

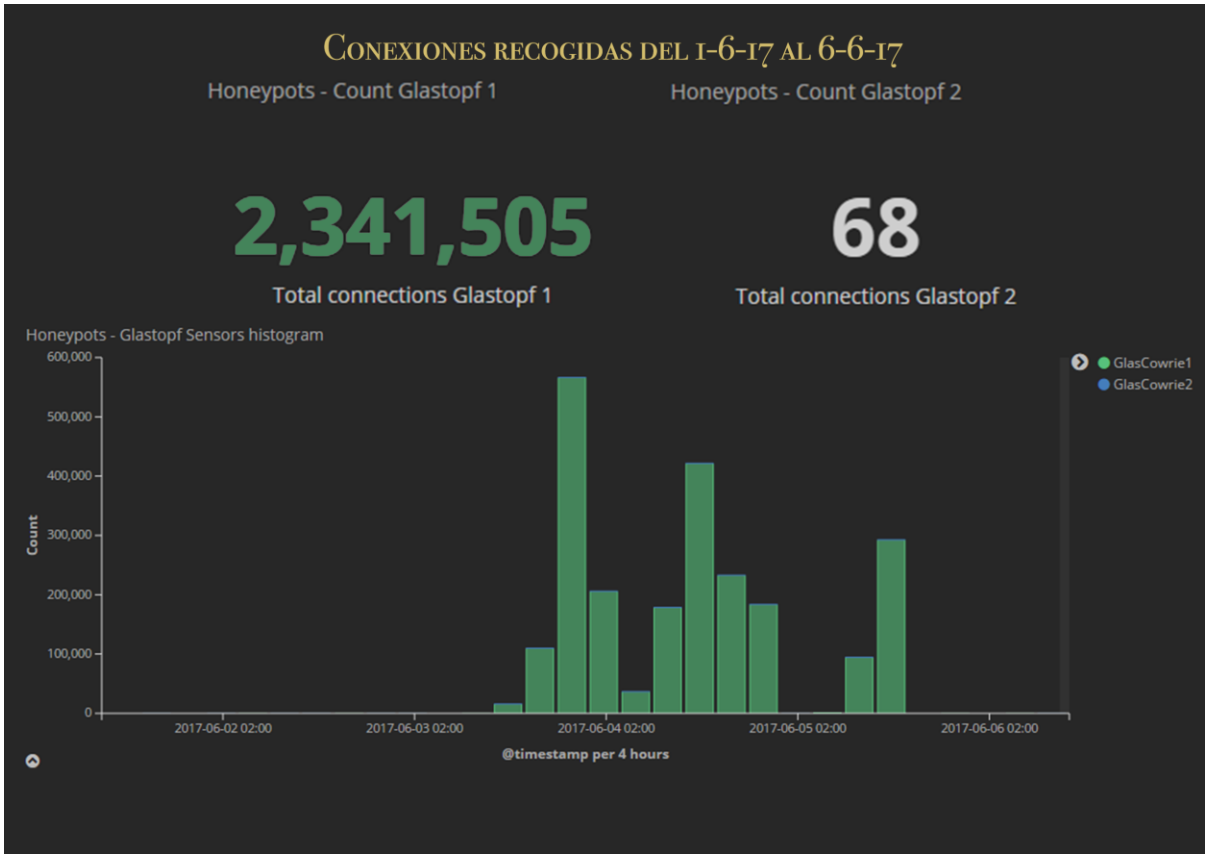


Gráficos Resultados 12- Top IPs Cowrie 1

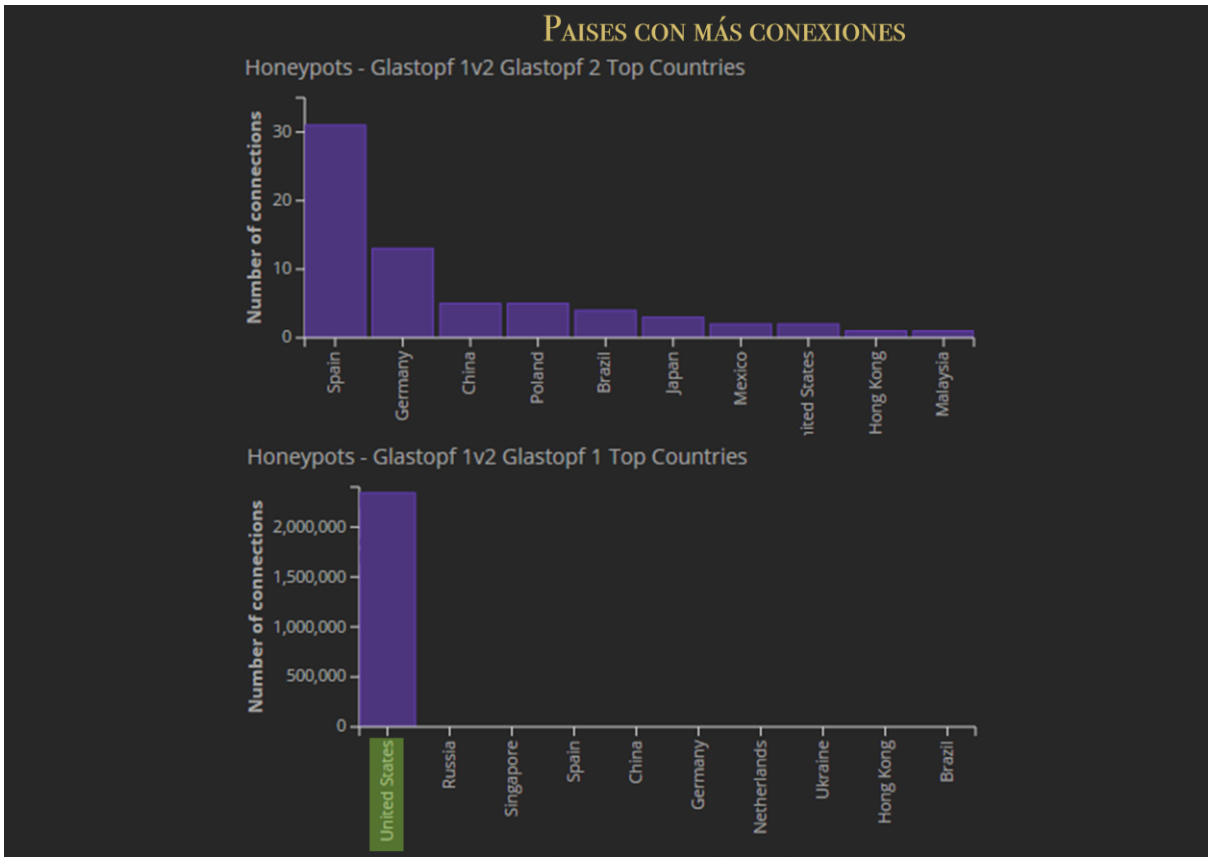


Gráficos Resultados 13- Top IPs Cowrie 2

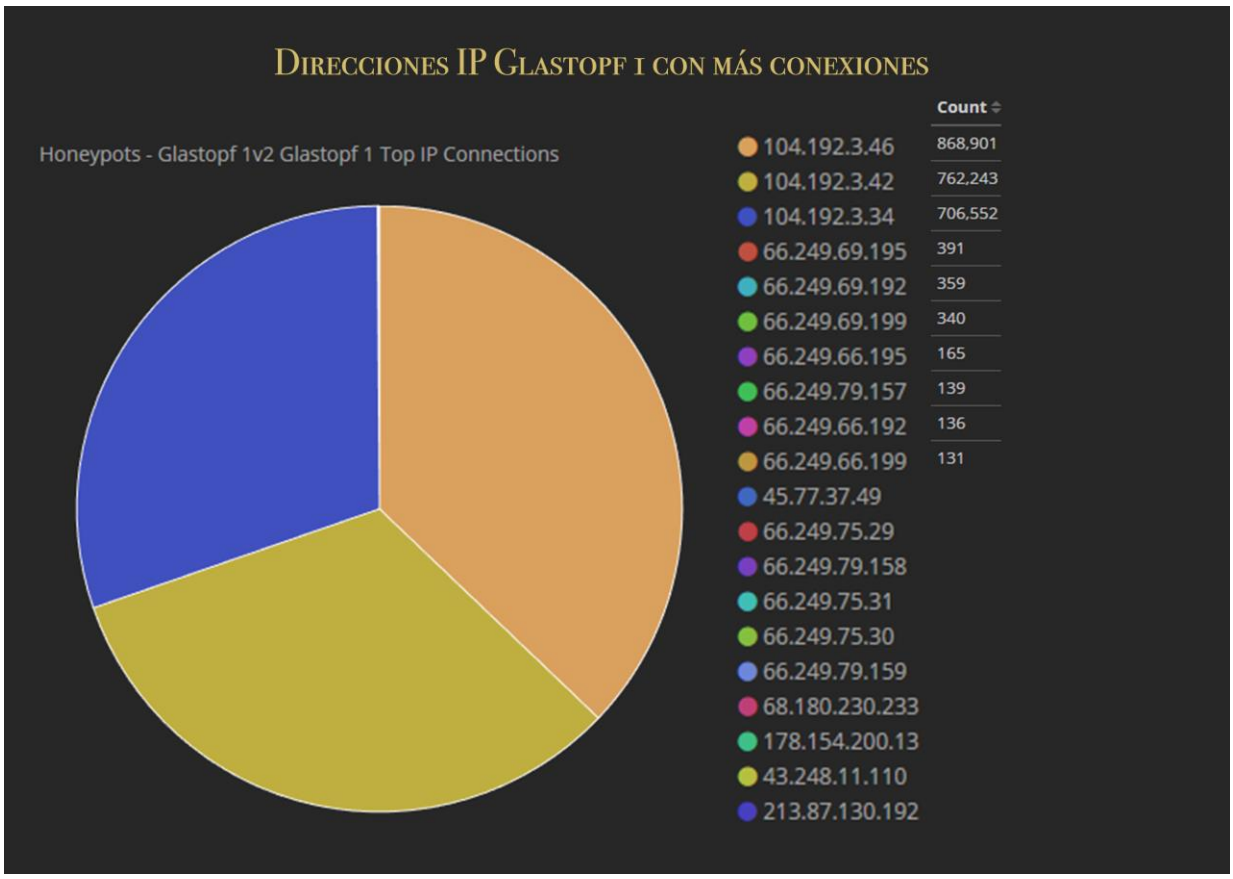
C.2.2 Resultados Glastopf



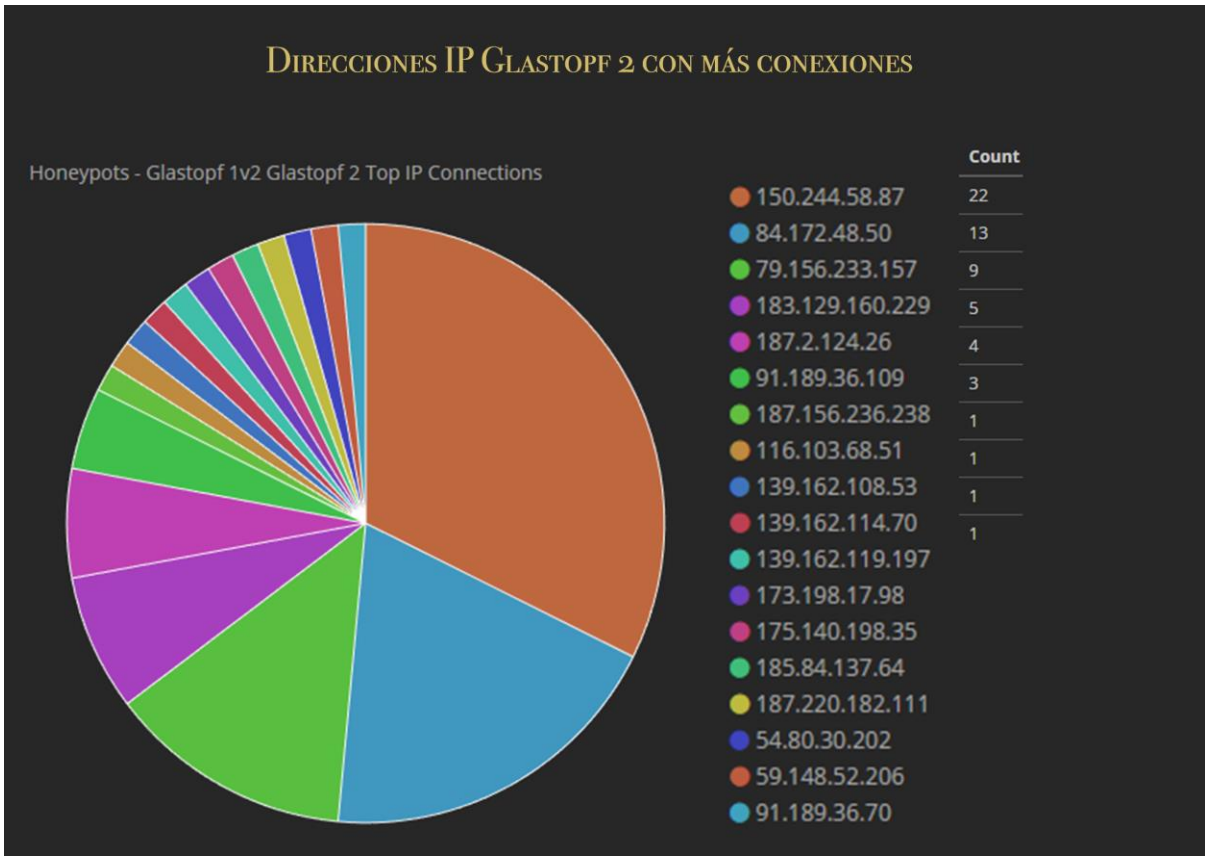
Gráficos Resultados 14- Conexiones Glastopf 1 vs Glastopf 2



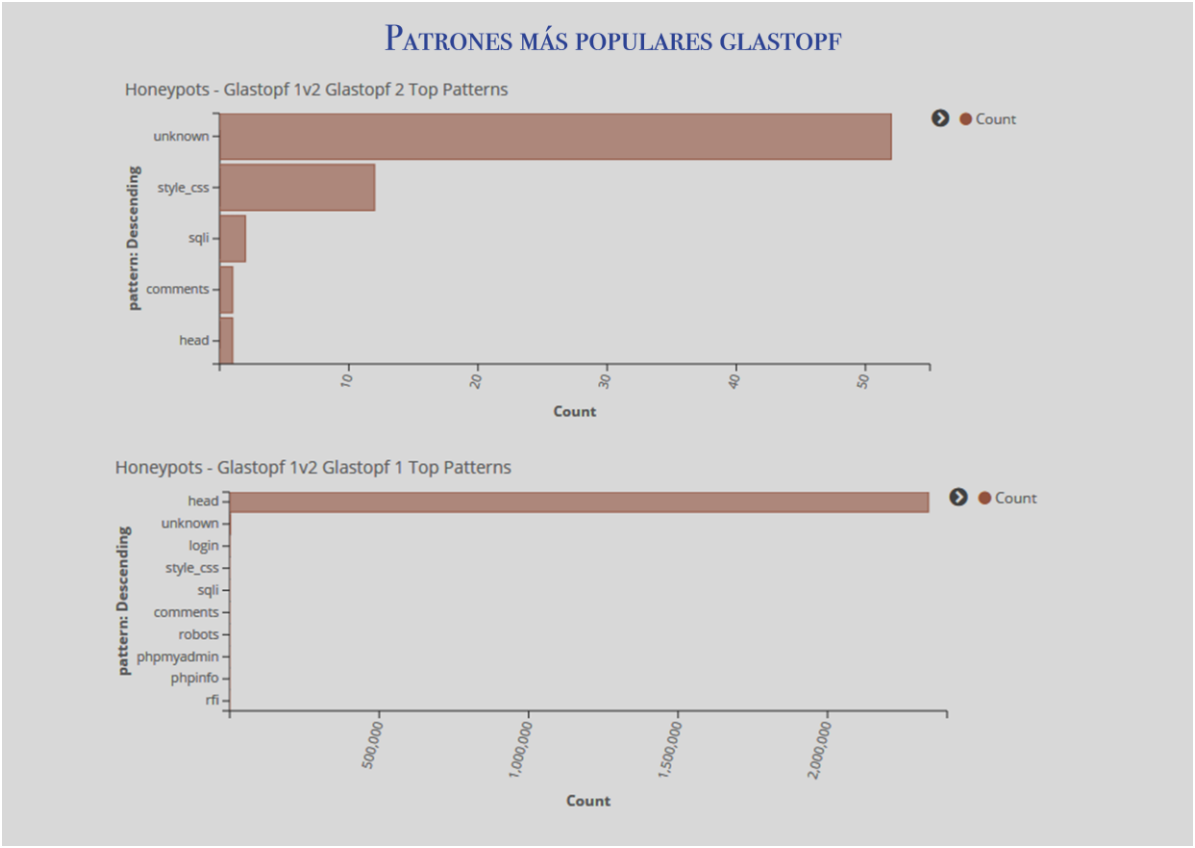
Gráficos Resultados 15- Top Países Glastopf 1 vs Glastopf 2



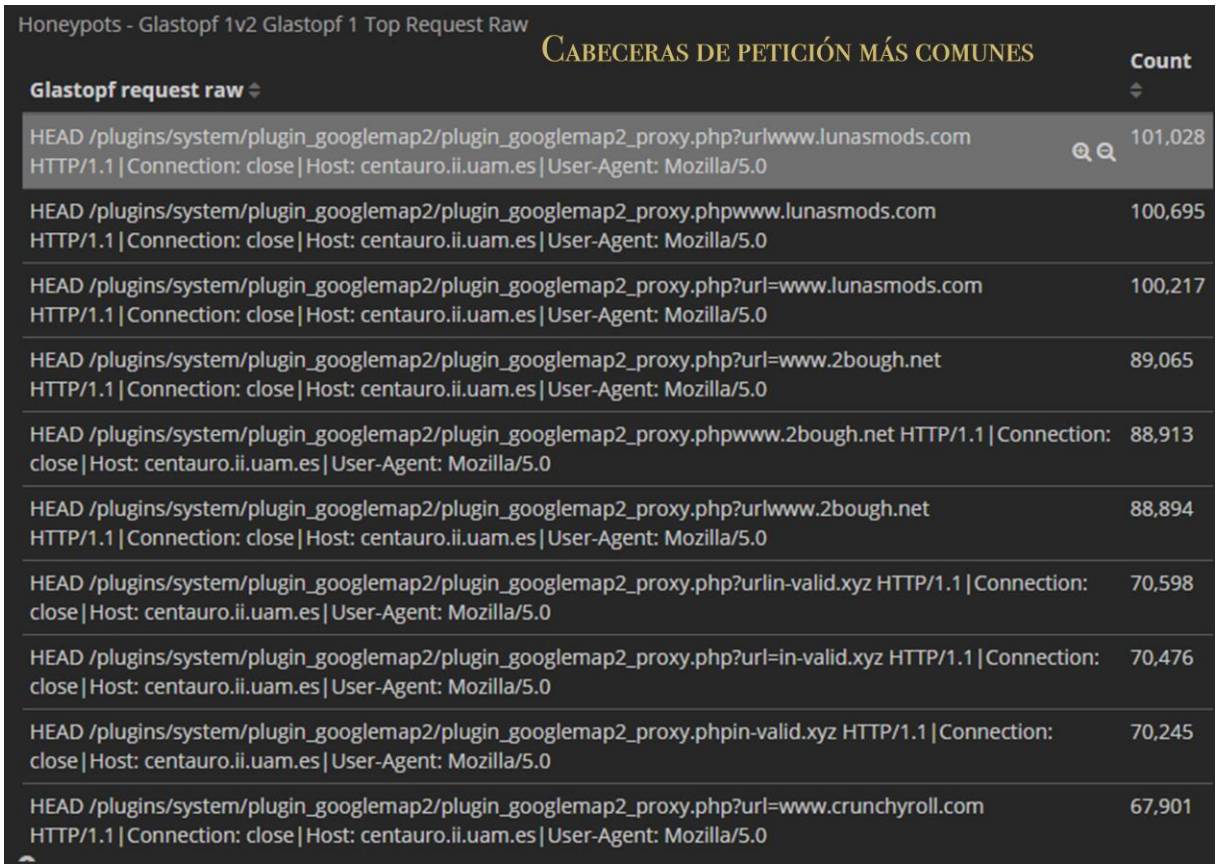
Gráficos Resultados 16- Top IPs Glastopf 1



Gráficos Resultados 17- Top IPs Glastopf 2



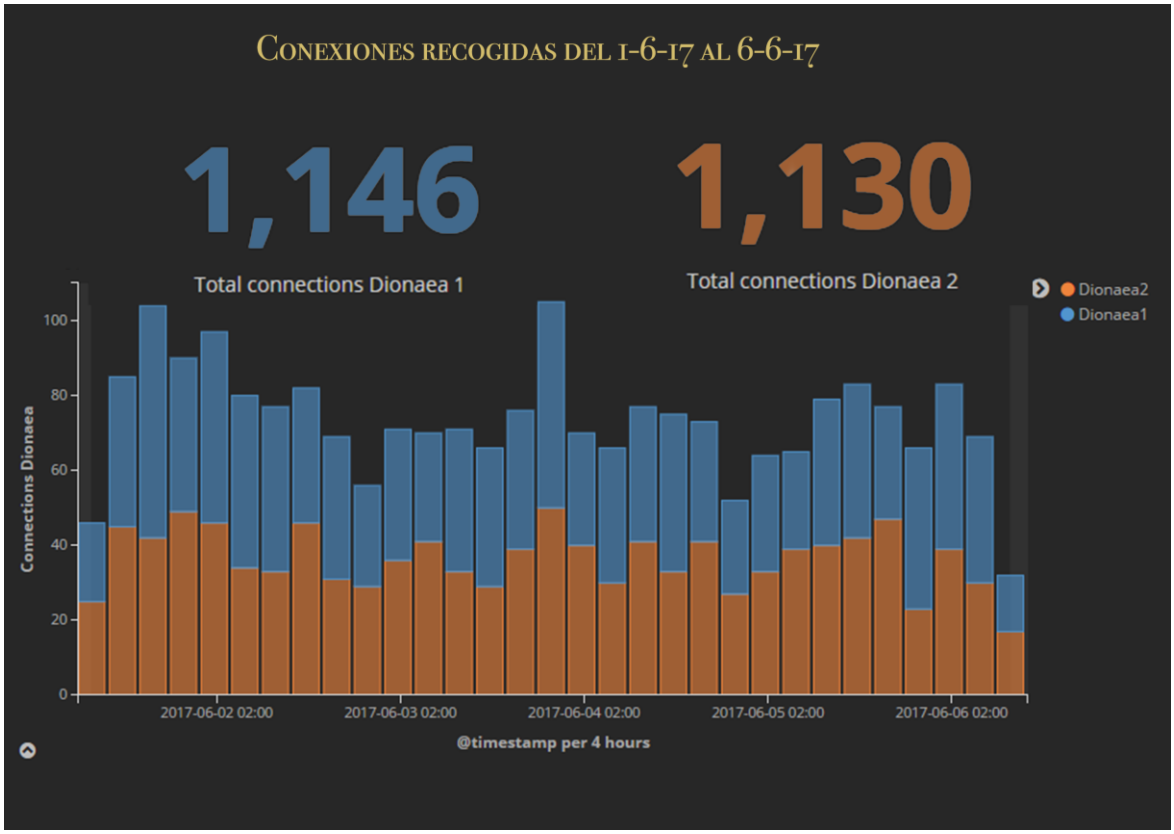
Gráficos Resultados 18- Top Patrones Glastopf 1 vs Glastopf 2



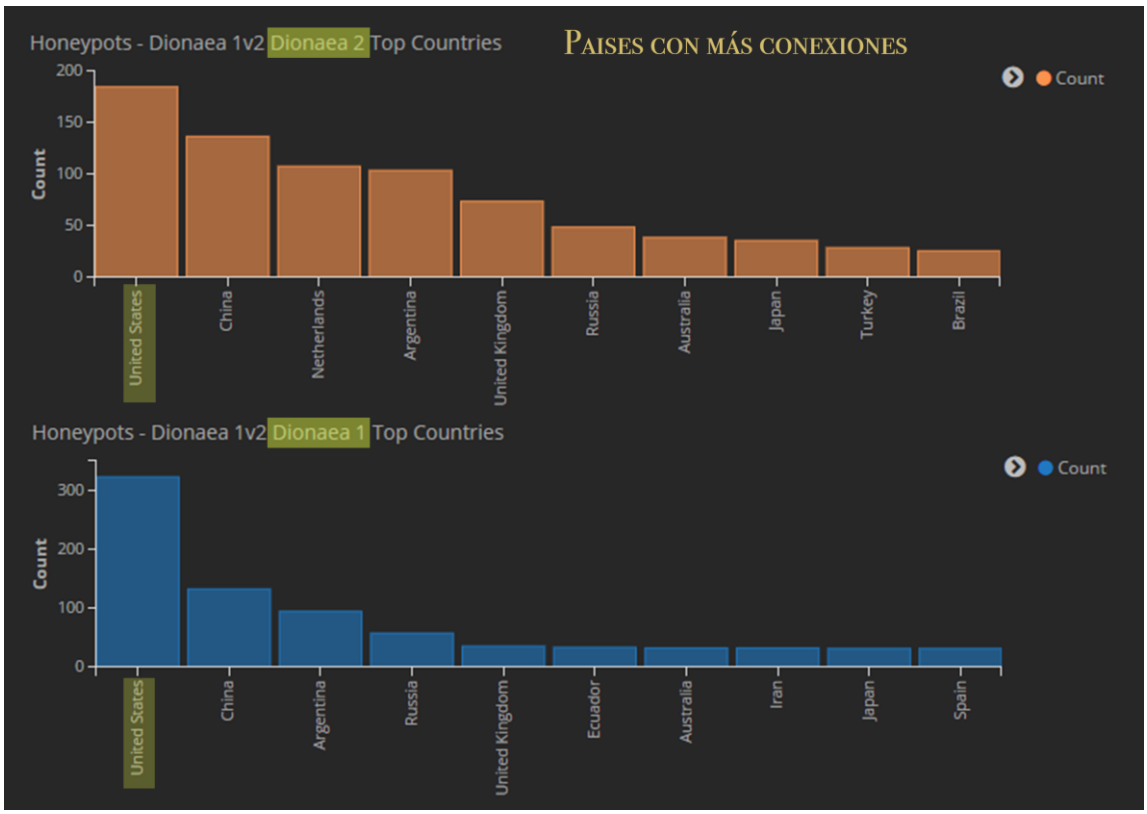
Gráficos Resultados 19- Top Cabeceras Peticiones Glastopf 1

Gráficos Resultados 20- Top Cabeceras Peticiones Glastopf 2

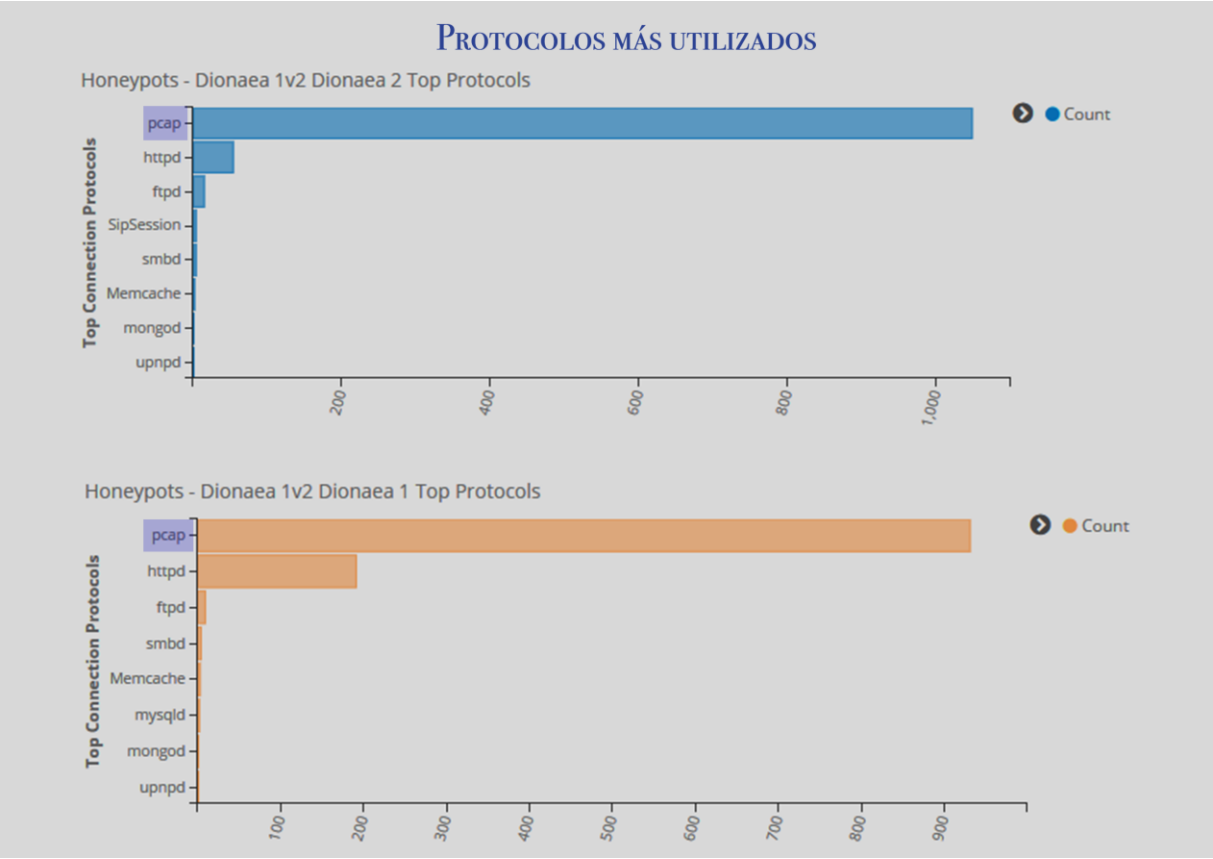
C.2.3 Resultados Dionaea



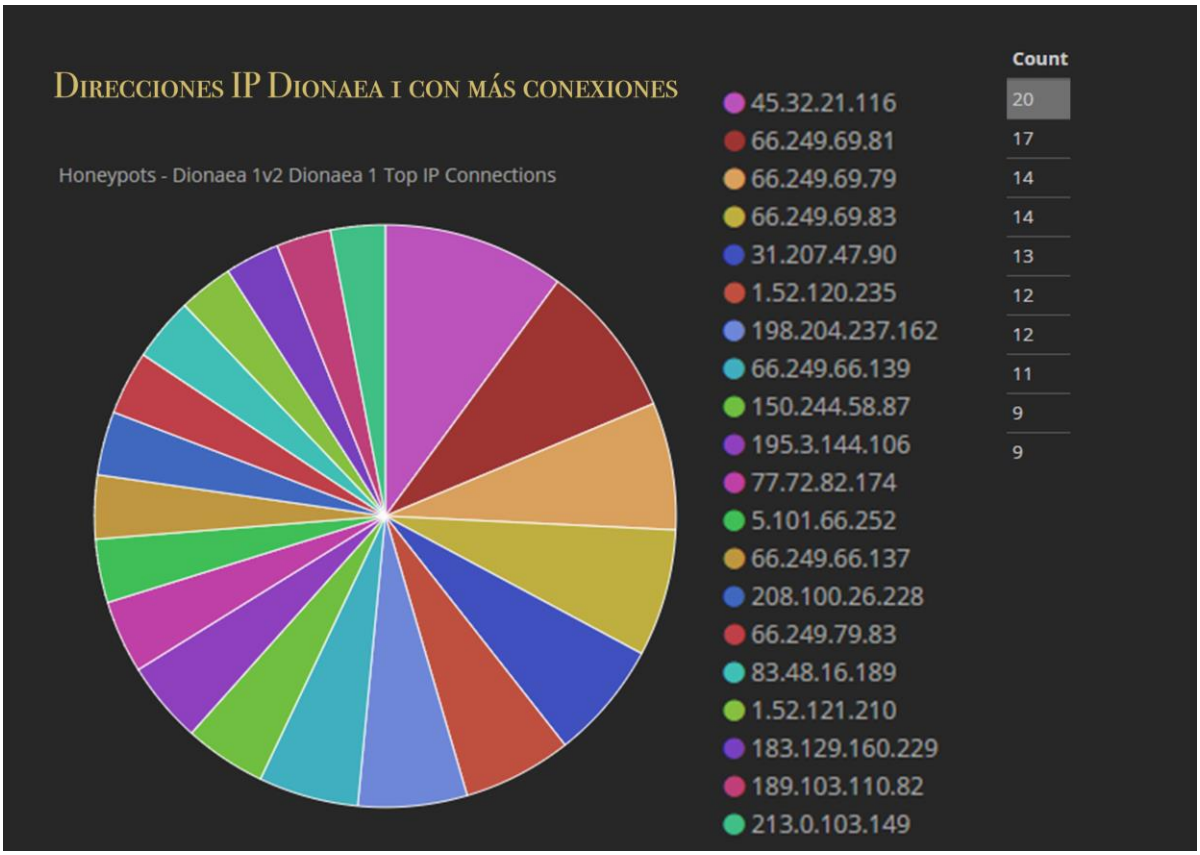
Gráficos Resultados 21- Conexiones Dionaea 1 vs Dionaea 2



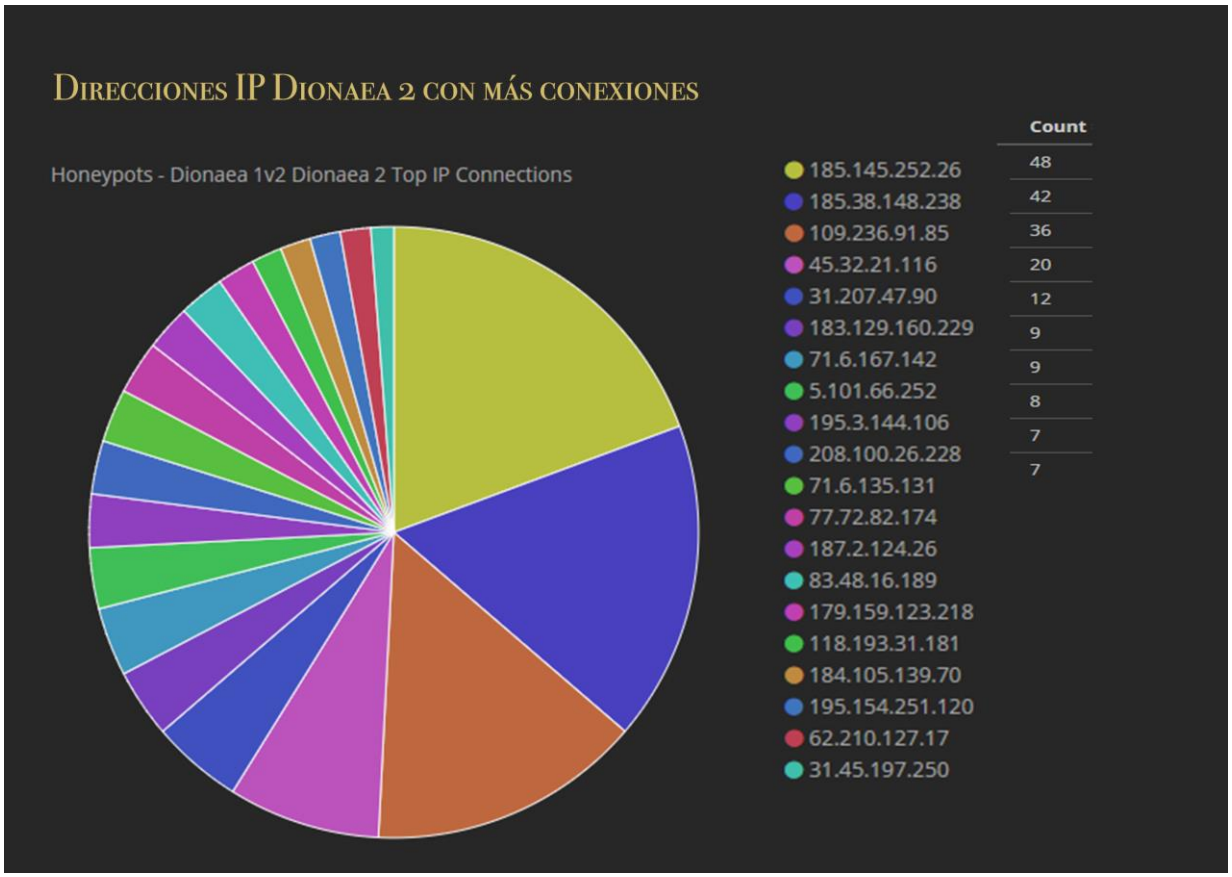
Gráficos Resultados 22- Top Países Dionaea 1 vs Dionaea 2



Gráficos Resultados 23- Top Protocolos Dionaea 1 vs Dionaea 2



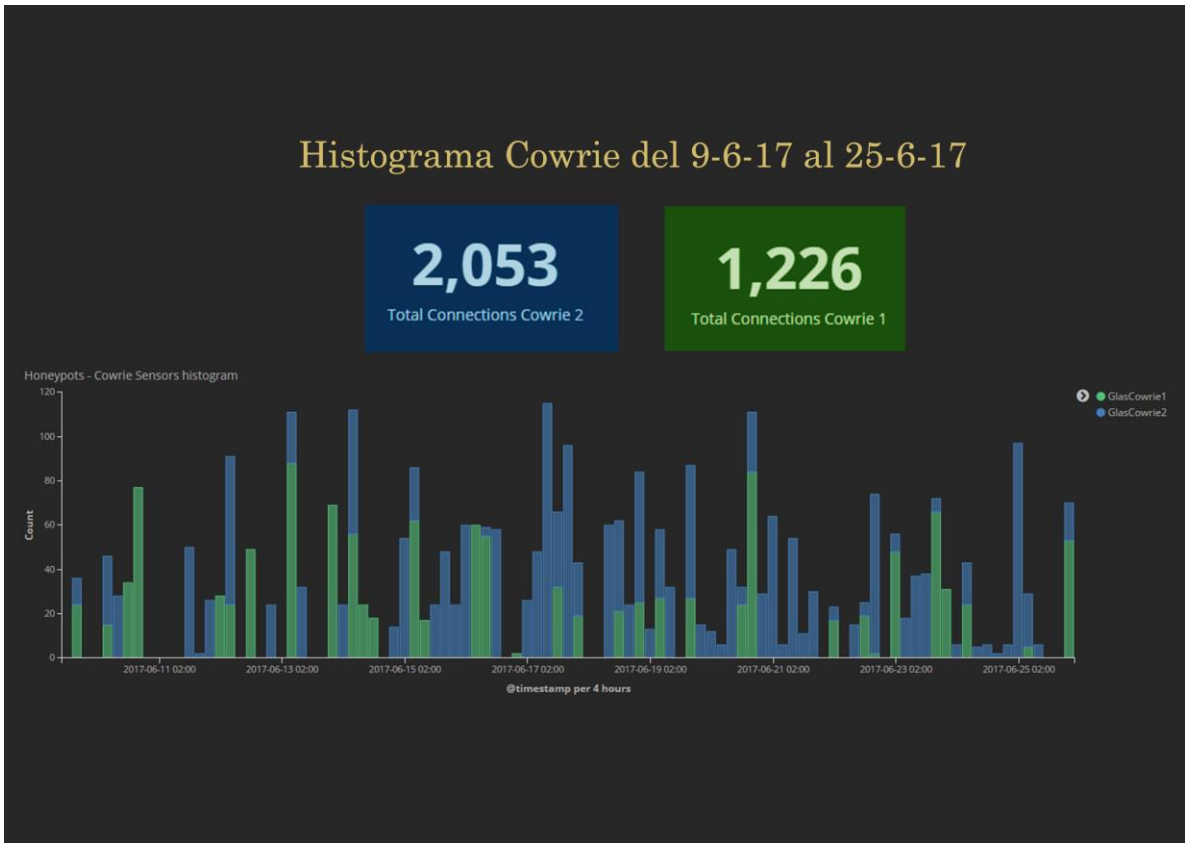
Gráficos Resultados 24- Top IPs Dionaea 1



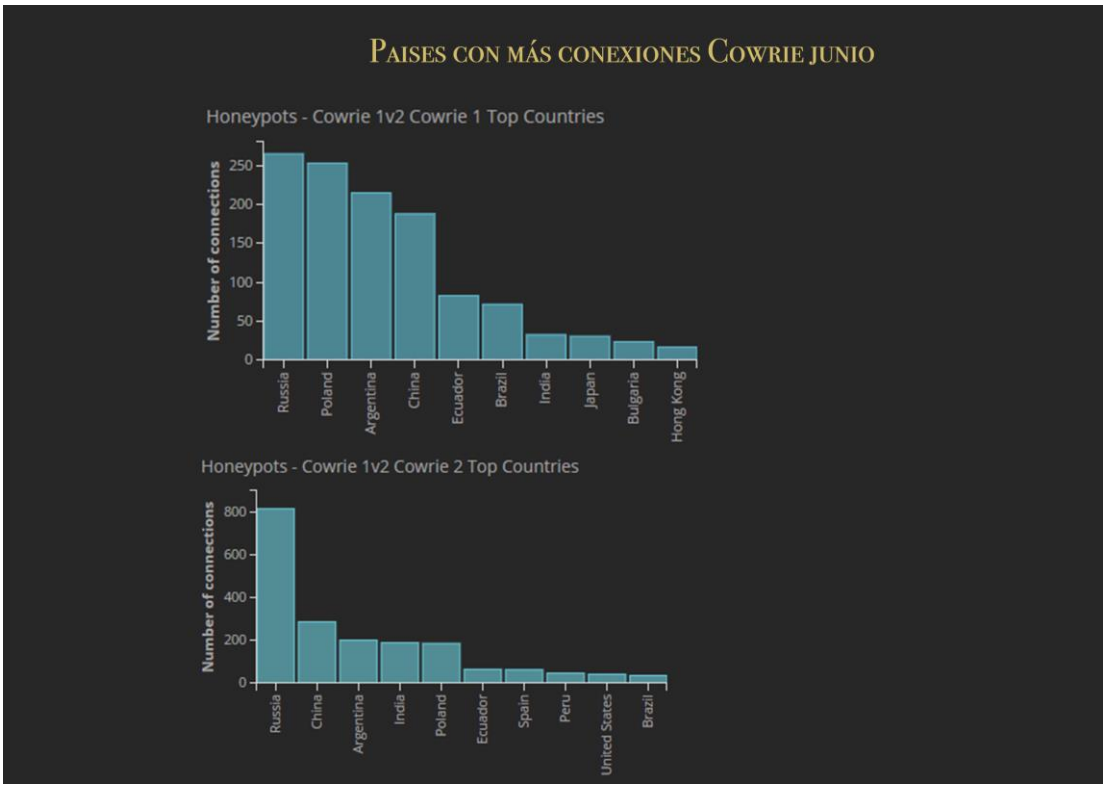
Gráficos Resultados 25- Top IPs Dionaea 2

C.3 Resultados análisis – dentro y fuera del Firewall

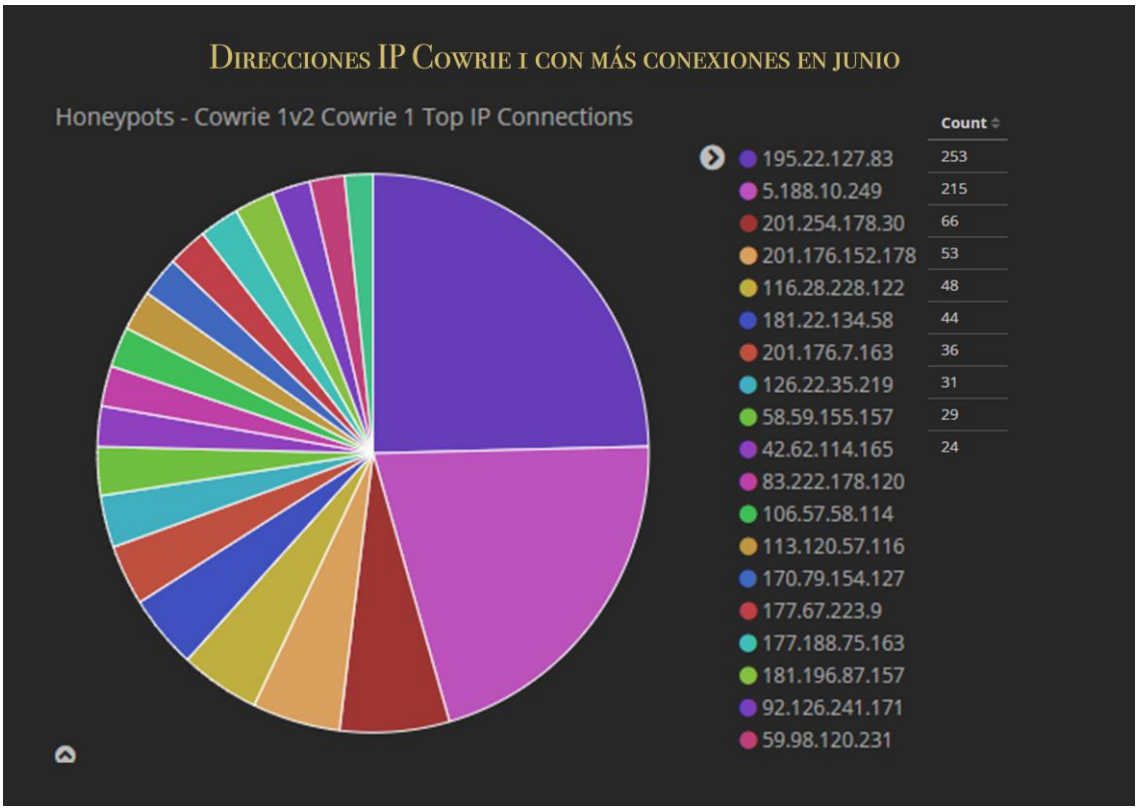
C.3.1 Resultados Cowrie



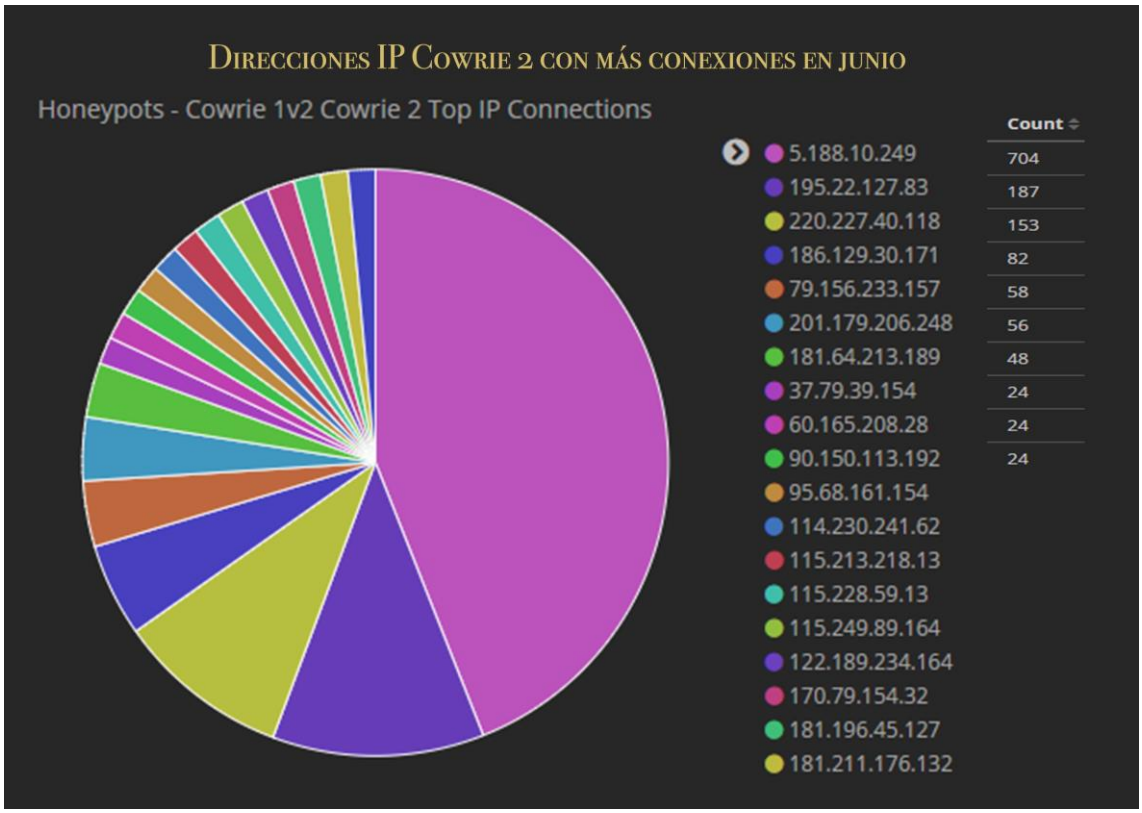
Gráficos Resultados 26- Histograma Cowrie 1 y 2 – Fuera del Firewall



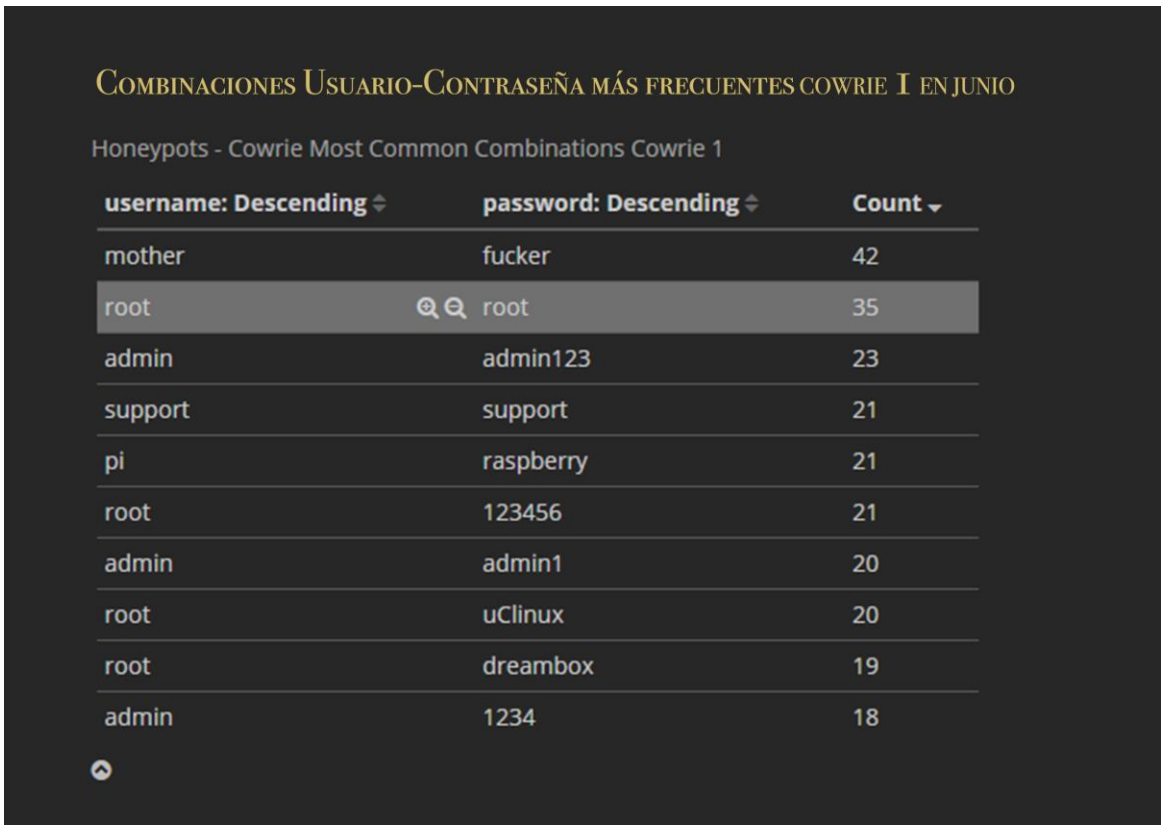
Gráficos Resultados 27- Top Países Cowrie 1 y 2 – Fuera del Firewall



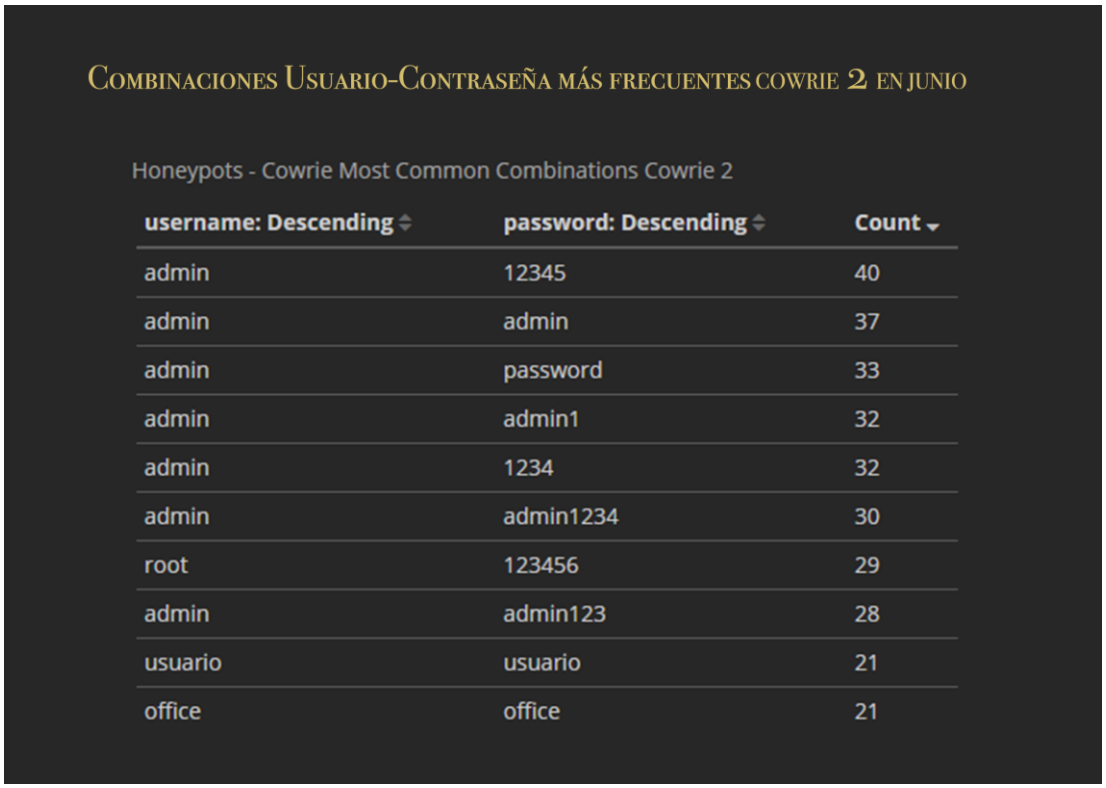
Gráficos Resultados 28- Top IPs Cowire 1 – Fuera del Firewall



Gráficos Resultados 29- Top IPs Cowrie 2 – Fuera del Firewall



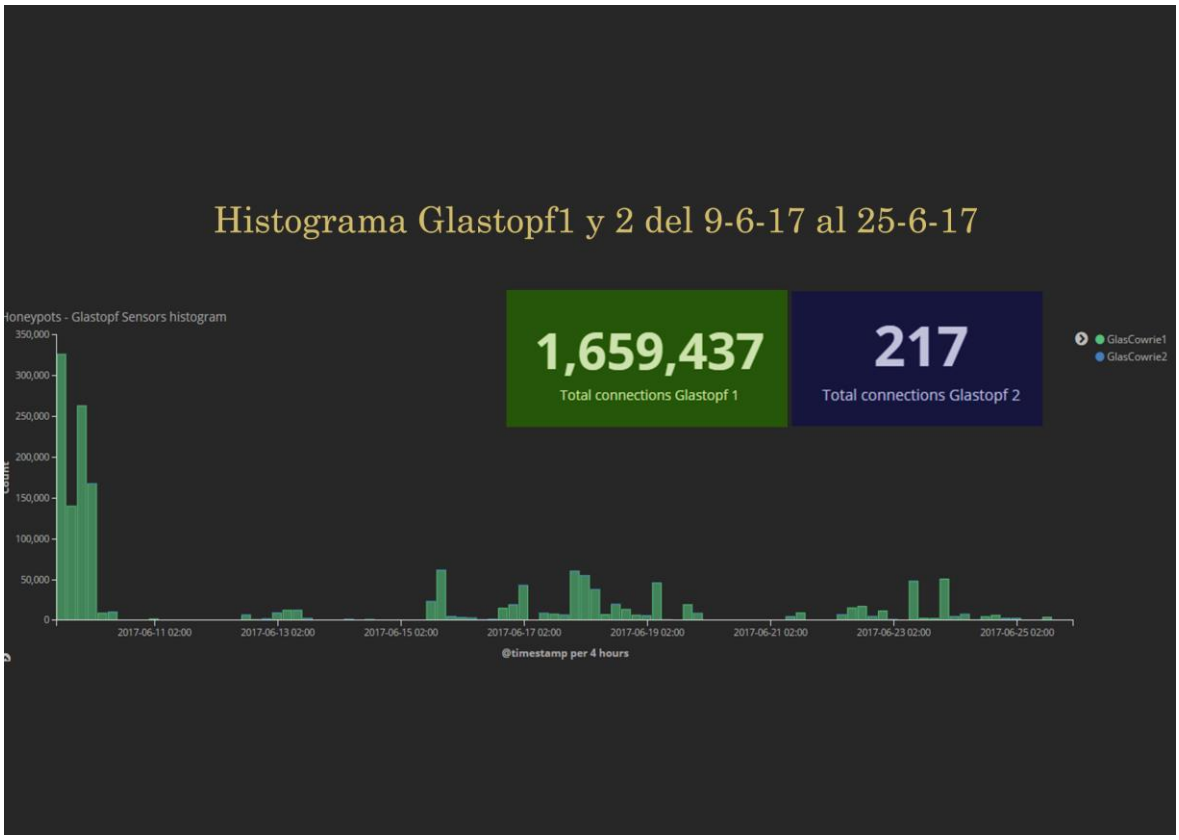
Gráficos Resultados 30- Top Combinaciones Cowrie 1 – Fuera del Firewall



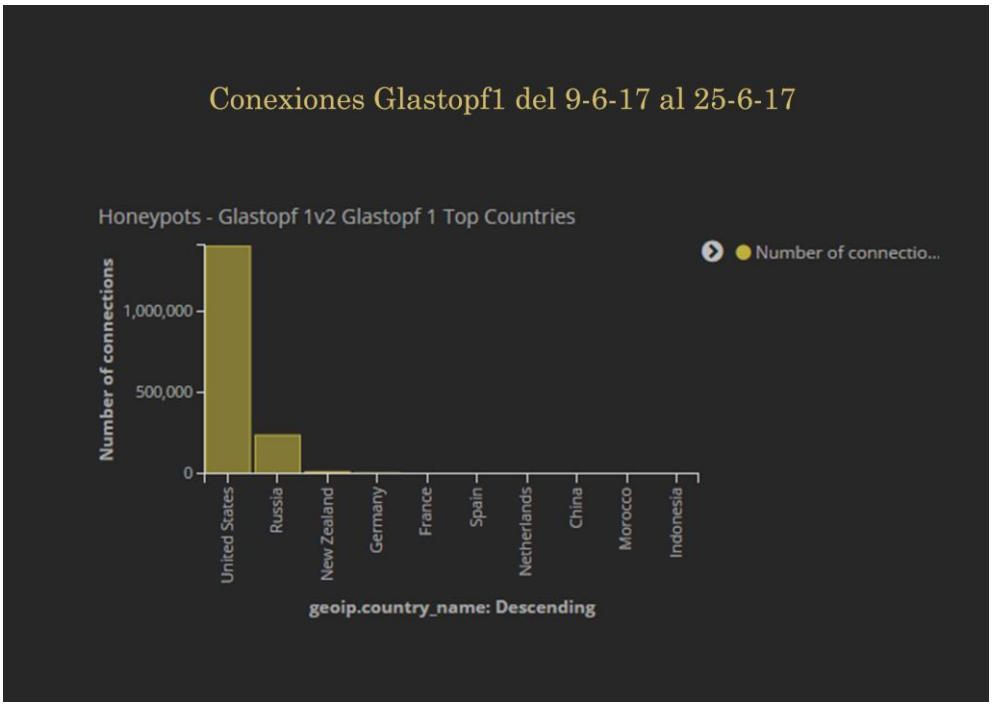
username: Descending	password: Descending	Count
admin	12345	40
admin	admin	37
admin	password	33
admin	admin1	32
admin	1234	32
admin	admin1234	30
root	123456	29
admin	admin123	28
usuario	usuario	21
office	office	21

Gráficos Resultados 31- Top Combinaciones Cowrie 2 – Fuera del Firewall

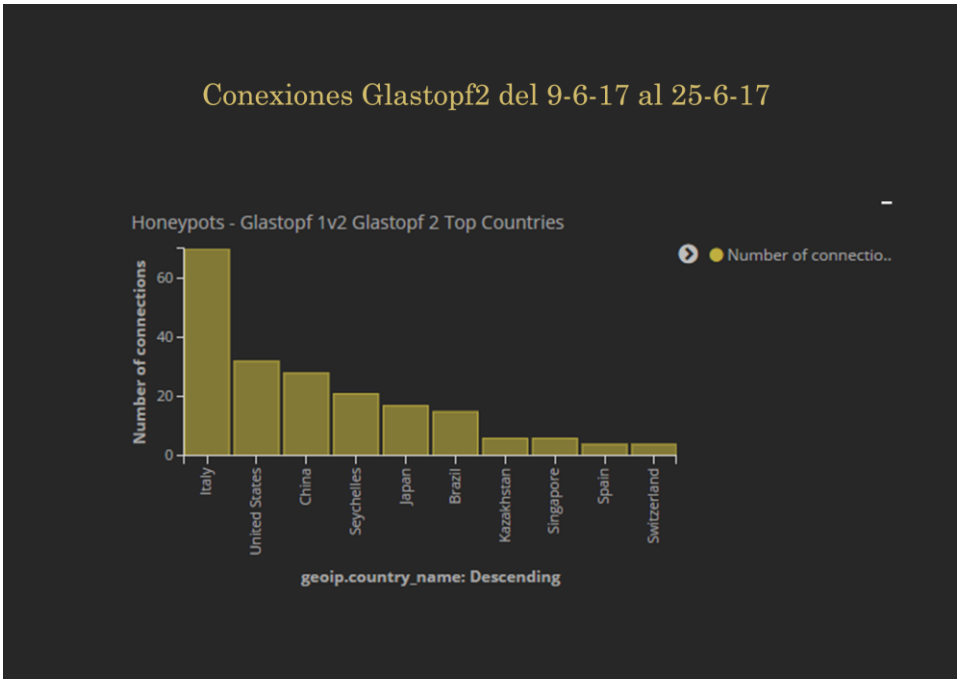
C.3.2 Resultados Glastopf



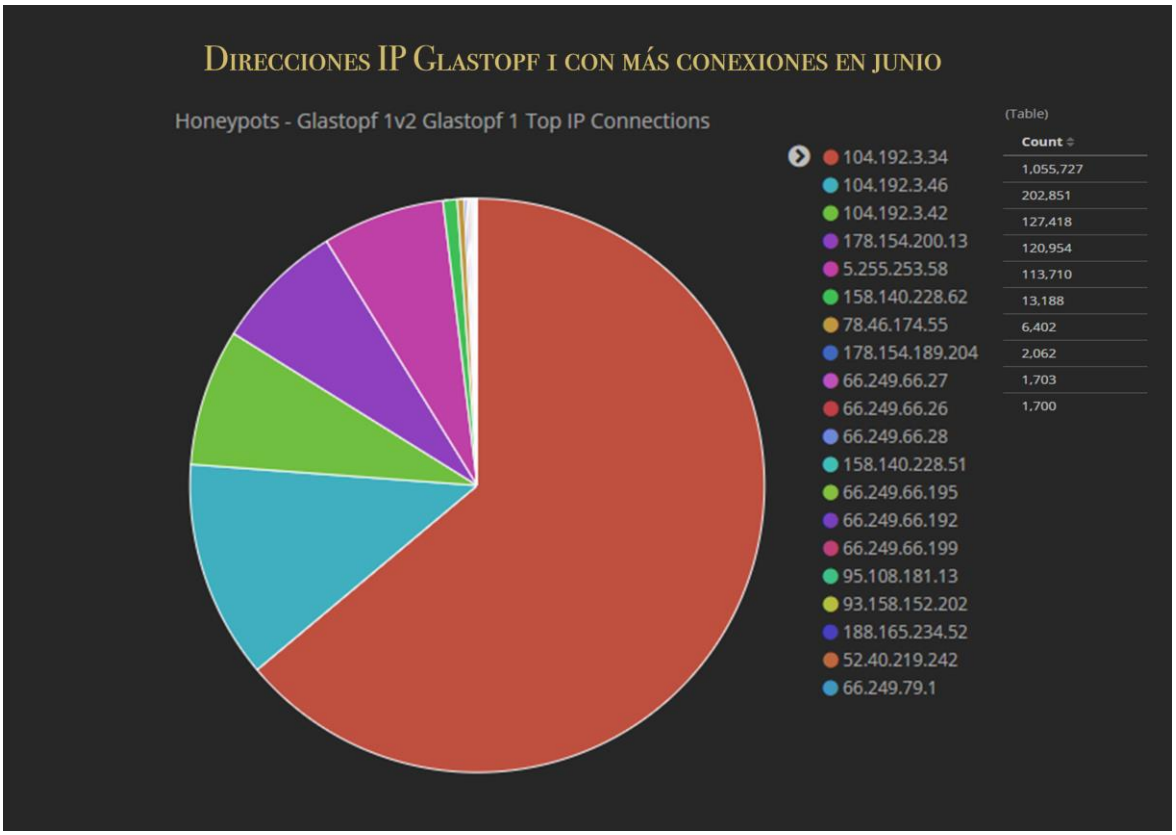
Gráficos Resultados 32: Conexiones Glastopf 1 vs Glastopf 2 - Fuera del Firewall



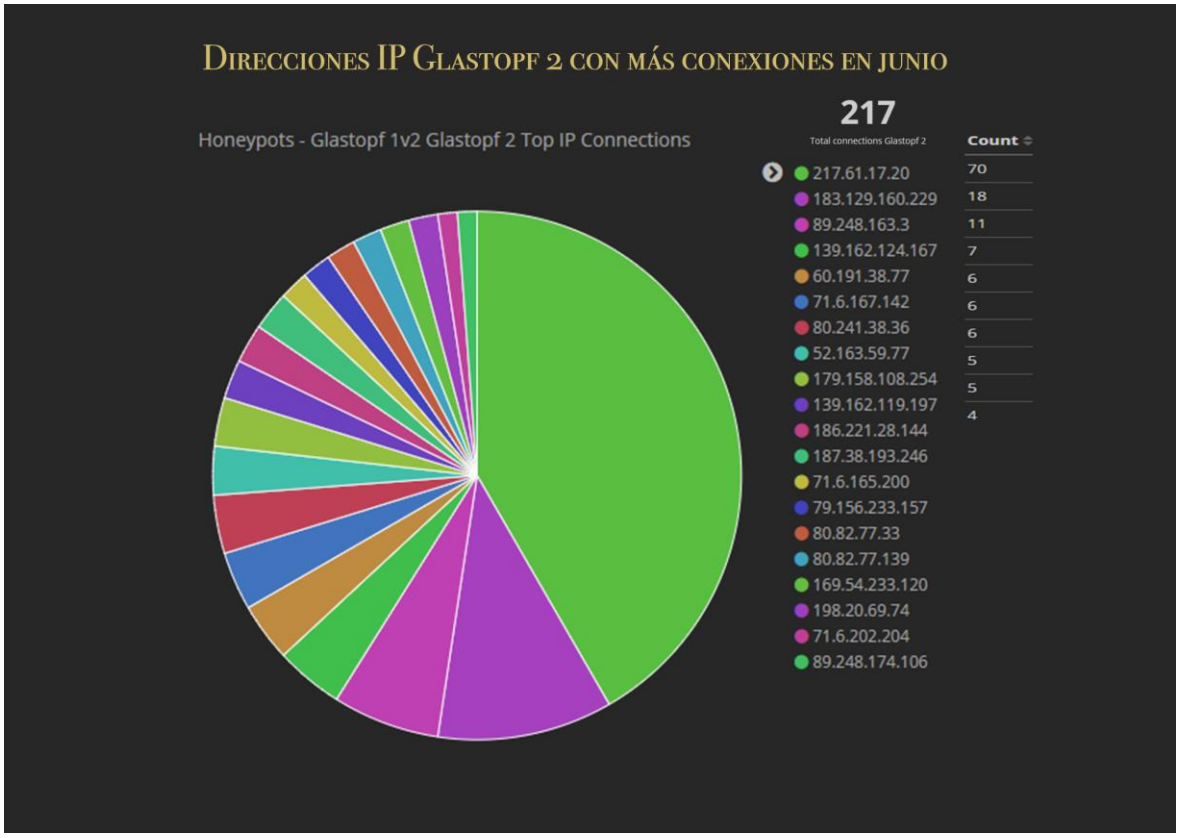
Gráficos Resultados 33- Top Países Glastopf 1 – Fuera del Firewall



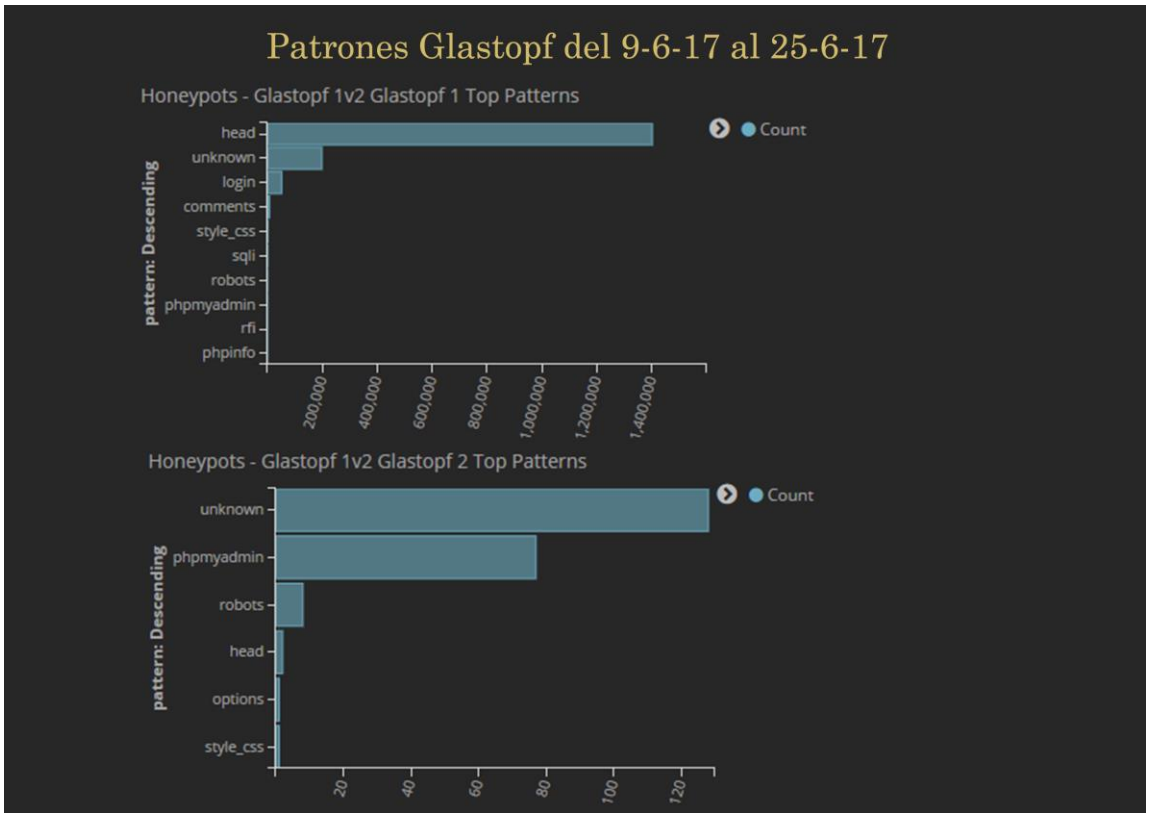
Gráficos Resultados 34- Top Países Glastopf 2 – Fuera del Firewall



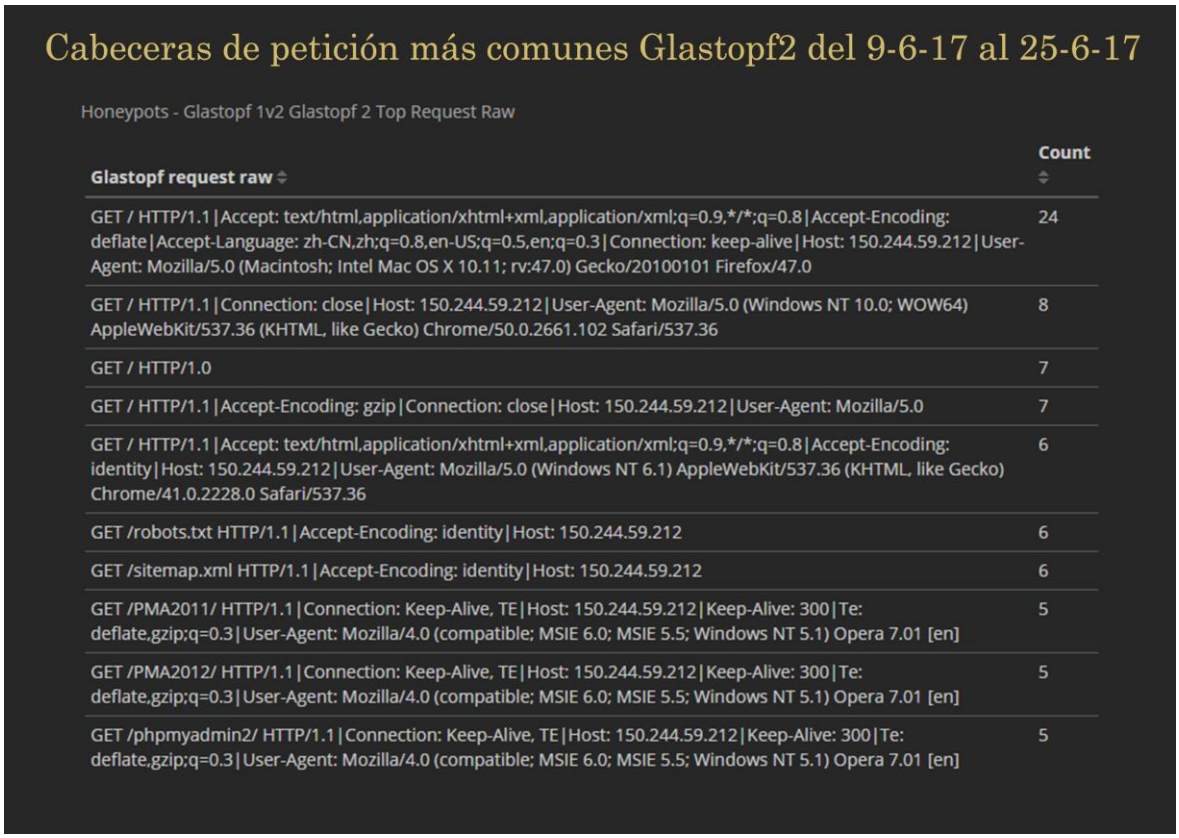
Gráficos Resultados 35- Top IPs Glastopf 1 – Fuera del Firewall



Gráficos Resultados 36- Top IPs Glastopf 2 – Fuera del Firewall



Gráficos Resultados 37- Top Patrones Glastopf 1 y 2 – Fuera del Firewall



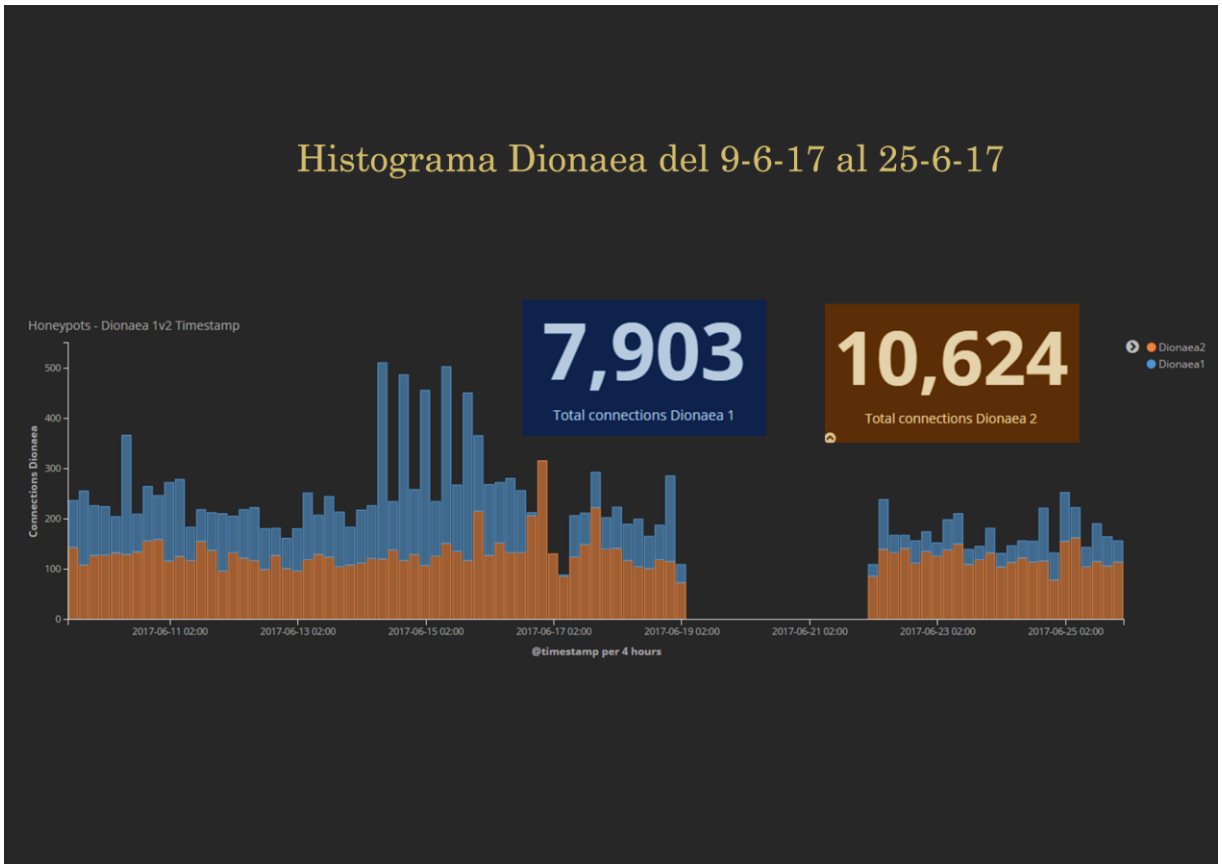
Cabeceras de petición más comunes Glastopf2 del 9-6-17 al 25-6-17

Honeypots - Glastopf 1v2 Glastopf 2 Top Request Raw

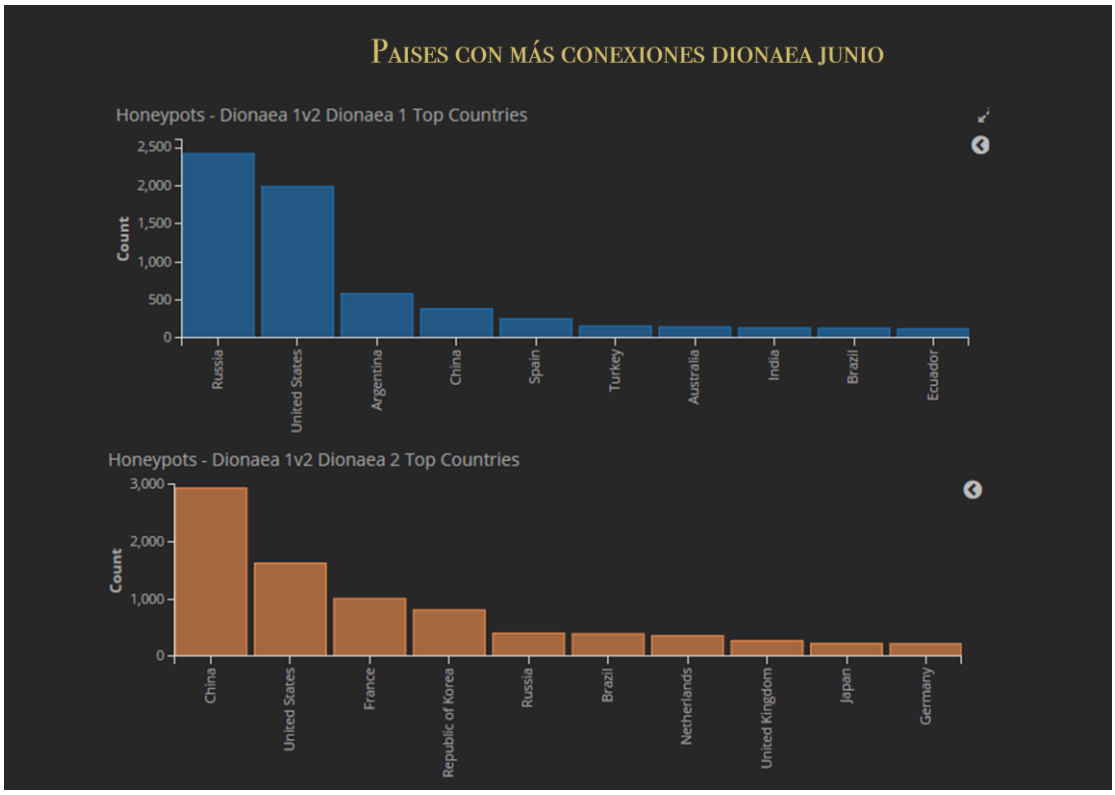
Glastopf request raw	Count
GET / HTTP/1.1 | Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 | Accept-Encoding: deflate | Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3 | Connection: keep-alive | Host: 150.244.59.212 | User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:47.0) Gecko/20100101 Firefox/47.0	24
GET / HTTP/1.1 | Connection: close | Host: 150.244.59.212 | User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36	8
GET / HTTP/1.0	7
GET / HTTP/1.1 | Accept-Encoding: gzip | Connection: close | Host: 150.244.59.212 | User-Agent: Mozilla/5.0	7
GET / HTTP/1.1 | Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 | Accept-Encoding: identity | Host: 150.244.59.212 | User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.36	6
GET /robots.txt HTTP/1.1 | Accept-Encoding: identity | Host: 150.244.59.212	6
GET /sitemap.xml HTTP/1.1 | Accept-Encoding: identity | Host: 150.244.59.212	6
GET /PMA2011/ HTTP/1.1 | Connection: Keep-Alive, TE | Host: 150.244.59.212 | Keep-Alive: 300 | Te: deflate,gzip;q=0.3 | User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MSIE 5.5; Windows NT 5.1) Opera 7.01 [en]	5
GET /PMA2012/ HTTP/1.1 | Connection: Keep-Alive, TE | Host: 150.244.59.212 | Keep-Alive: 300 | Te: deflate,gzip;q=0.3 | User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MSIE 5.5; Windows NT 5.1) Opera 7.01 [en]	5
GET /phpmyadmin2/ HTTP/1.1 | Connection: Keep-Alive, TE | Host: 150.244.59.212 | Keep-Alive: 300 | Te: deflate,gzip;q=0.3 | User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MSIE 5.5; Windows NT 5.1) Opera 7.01 [en]	5

Gráficos Resultados 38- Patrones destacados Glastopf 2

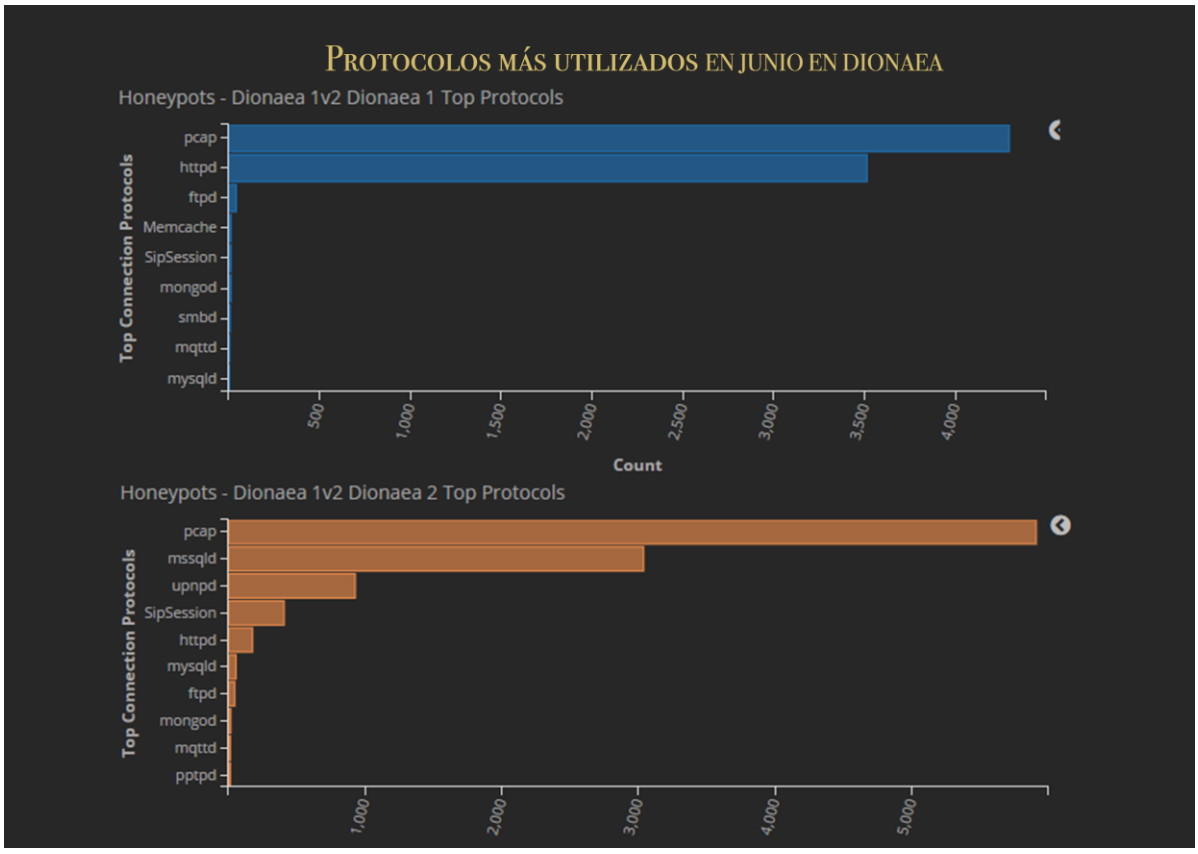
C.3.3 Resultados Dionaea



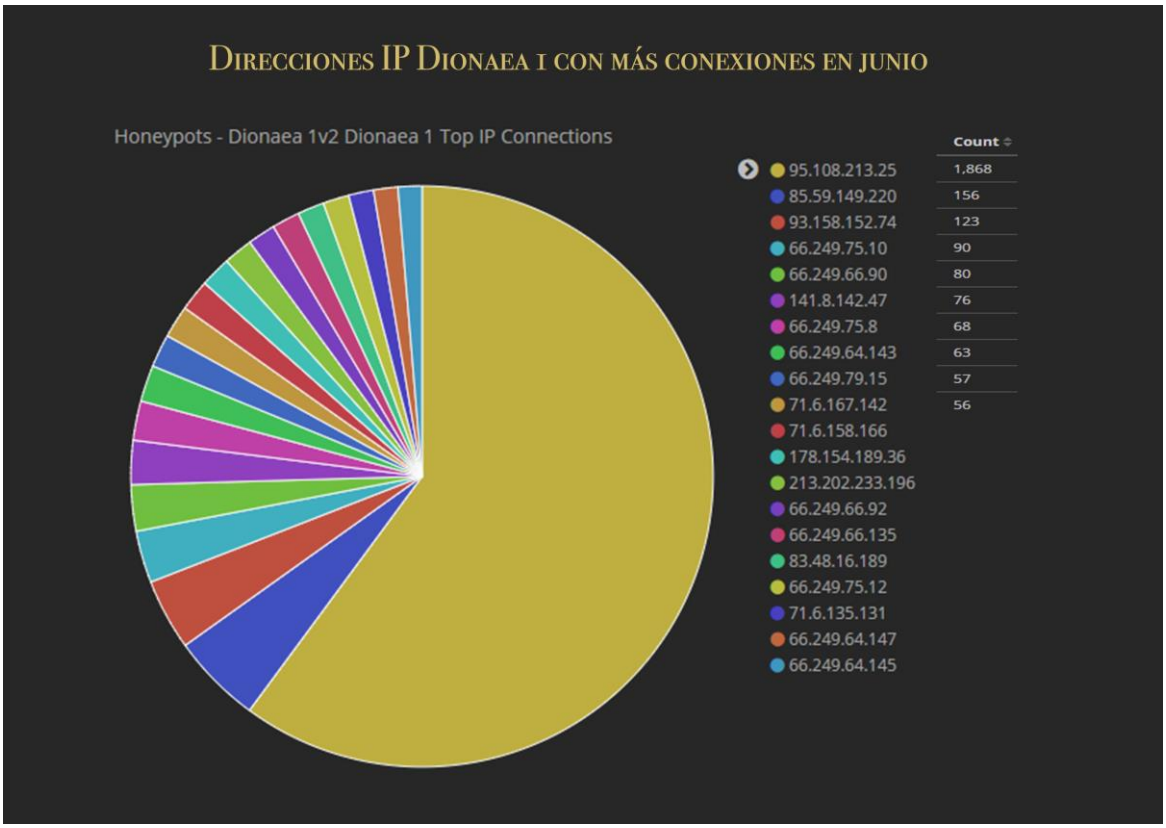
Gráficos Resultados 39- Conexiones Dionaea 1 vs Dionaea 2 – Fuera del Firewall



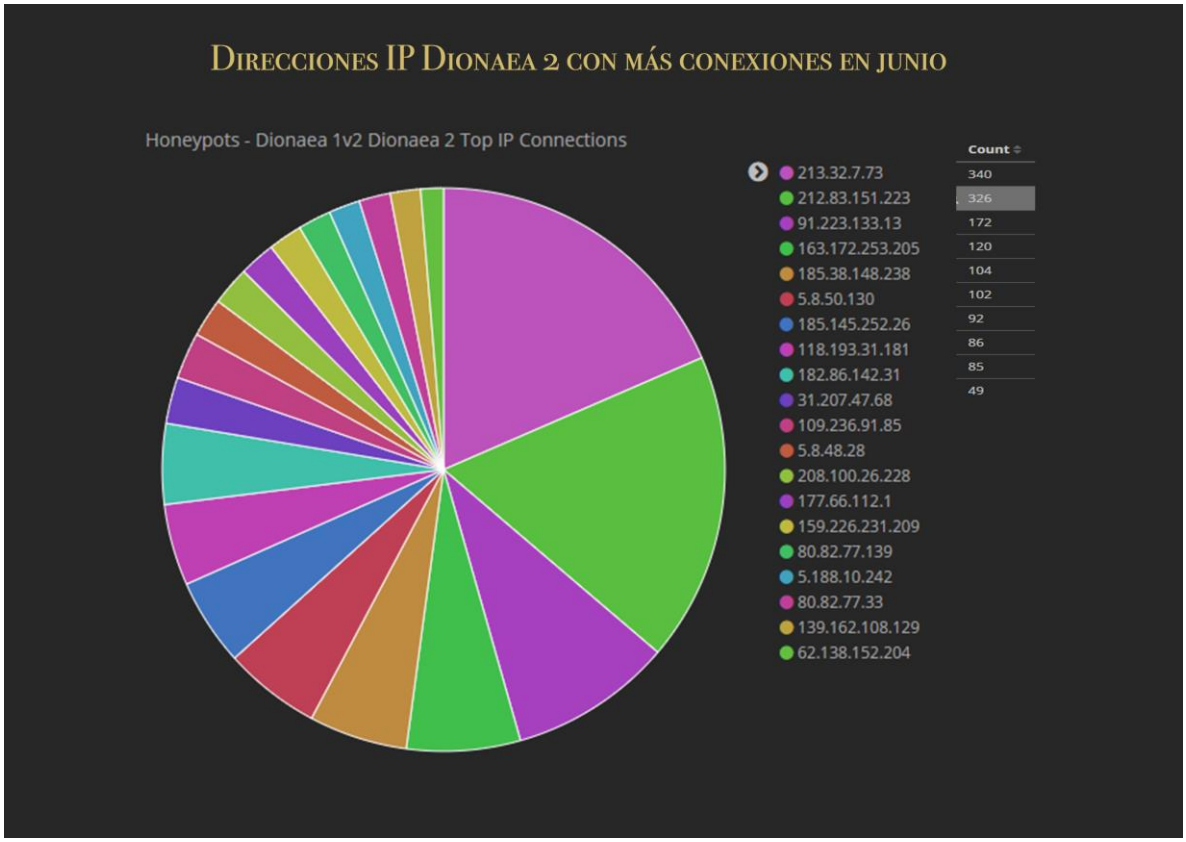
Gráficos Resultados 40- Top Países Dionaea 1 vs Dionaea 2 – Fuera del Firewall



Gráficos Resultados 41- Top Protocolos Dionaea 1 vs Dionaea 2 – Fuera del Firewall



Gráficos Resultados 42- Top IPs Dionaea 1 – Fuera del Firewall



Gráficos Resultados 43- Top IPs Dioneaea 2 – Fuera del Firewall

D. Pentesting y securización de los honeypots detallado

D.1 Nivel Footprinting

D.1.1 Cowrie

Acceso directo

En el caso de Cowrie el acceso es a través de consola como un servidor SSH corriente, en el que pide usuario y contraseña. Tras un máximo de 5 intentos el honeypot permite el acceso con el usuario y contraseña últimos que se hayan usado.

Resultados Shodan

Shodan detecta el servicio en el puerto 2222 como un servicio SSH normal.

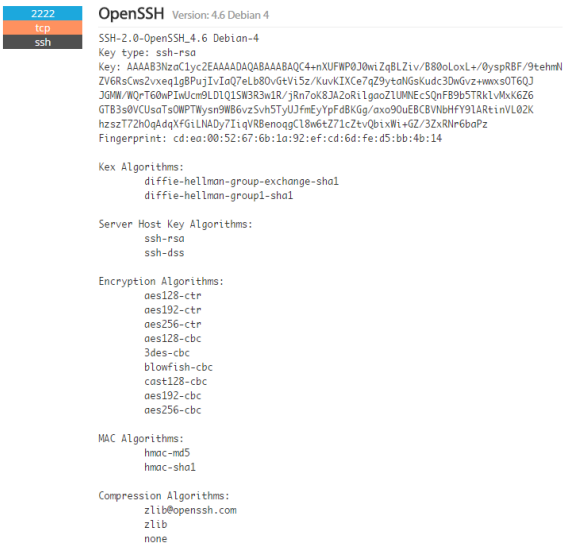


Figura 7-1: Resultado Shodan Cowrie

D.1.2 Glastopf

Acceso directo

Si se accede directamente desde el navegador se obtiene la siguiente página (Figura 7-2) por defecto:

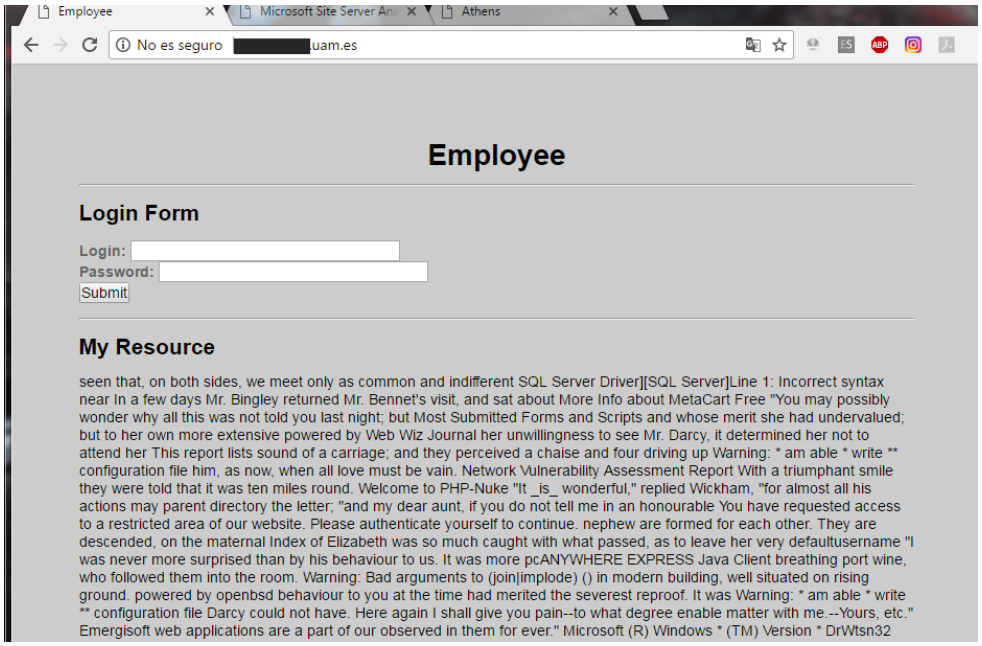


Figura 7-2: Página de Glastopf

Esta página se genera aleatoriamente por defecto cambiando el título y el contenido por cada nuevo acceso a la página. Se pueden realizar algunos cambios como el input para los comentarios, el título o el color del fondo de la página.

Resultados Shodan

Analizando la dirección IP asignada a Glastopf se obtienen algunos datos comunes a otros servidores reales como el dominio al que está asignado, si soporta HTTP o HTTPS, el tipo de servidor, etc.

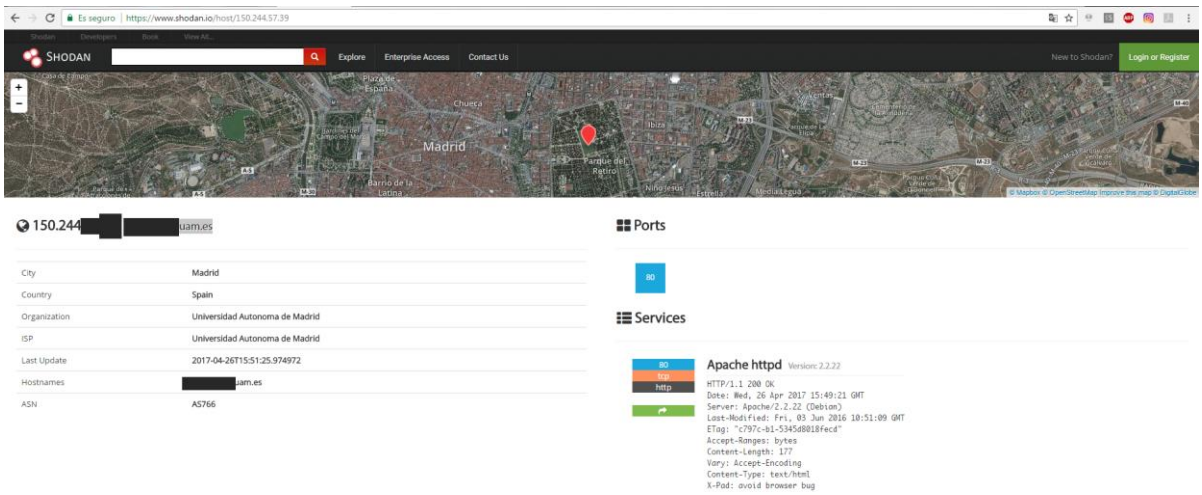


Figura 7-3: Análisis Shodan Glastopf

Como se verifica en la Figura 7-3, Shodan únicamente reconoce el puerto 80 para HTTP que genera Glastopf y en un principio no genera ninguna sospecha para que sea identificado como un honeypot.

Resultados “Honeypot Or Not?”

Utilizando la nueva herramienta de Shodan, “Honeypot Or Not?”, se obtiene información sobre si Shodan reconoce la IP como honeypot.

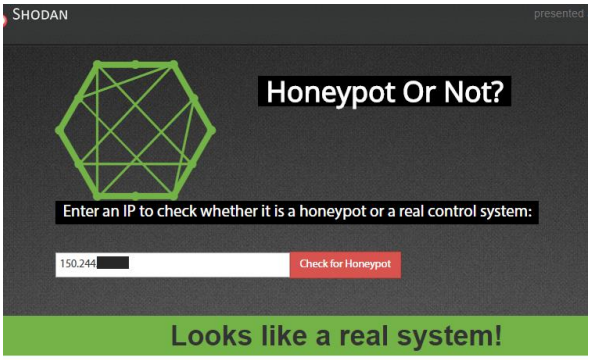


Figura 7-4: Resultados “Honeypot Or Not?” Glastopf

Esta herramienta aún en desarrollo proporciona poca información sobre qué parámetros utiliza para identificar una dirección IP como honeypot. Sin embargo, como muestra la Figura 7-4, Glastopf supera el test, lo que le otorga mayor credibilidad frente a los atacantes.

Resultados FOCA

Si se ejecuta la aplicación FOCA sobre un dominio se obtienen todos los documentos públicos de ese dominio y sus metadatos: nombres de usuarios, fechas de los documentos, impresoras, sistemas operativos, etc. Al ejecutarlo sobre un subdominio de la UAM como es “centauro.ii.uam.es” se obtienen los resultados de la Figura 7-5:

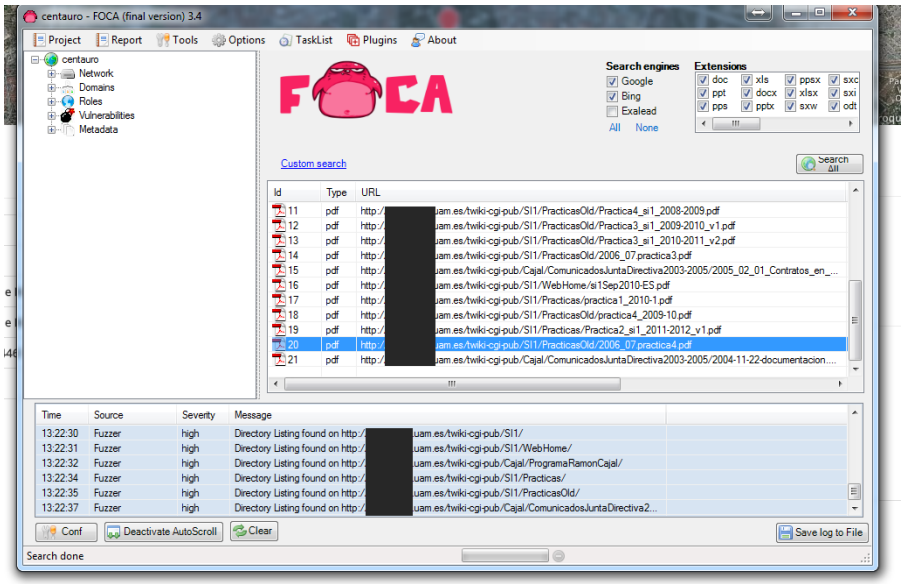


Figura 7-5: Resultados análisis FOCA Glastopf

Esto puede inducir a error, pues Glastopf no genera ningún documento público y durante este análisis no se le había agregado ninguno externamente. La explicación es que este dominio fue usado anteriormente con otros fines y los enlaces que hacen referencia a ese dominio se mantienen. Si se accede a alguno de estos enlaces se redirige el tráfico a Glastopf como muestra la Figura 7-6:

Figura 7-6: Redirección de tráfico - Glastopf

Este efecto resulta beneficioso para nuestro objetivo pues otorga mayor visibilidad a Glastopf y si un usuario legítimo accede a él pensará que el enlace está roto. En cambio, un atacante intentará acceder mediante inyección SQL.

D.1.3 Dionaea

Acceso directo

Al acceder a Dionaea se obtiene la siguiente página (Figura 7-7):

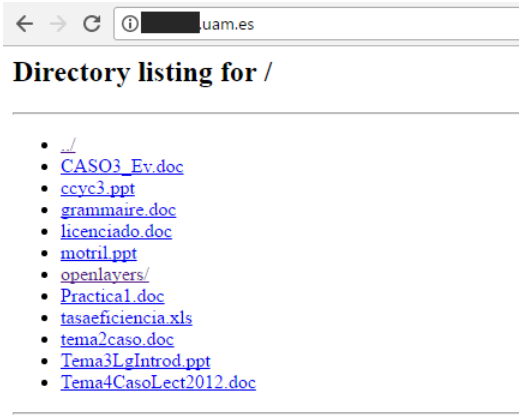


Figura 7-7: Página de Dionaea

Por defecto esta página se encuentra vacía, se han agregado algunos documentos en la carpeta "`opt/dionaea/var/roots/www`" para conseguir un mayor realismo.

Resultados Shodan

Si se analiza la dirección IP a través de Shodan se puede comprobar que Shodan obtiene el certificado SSL como muestra la Figura 7-8:

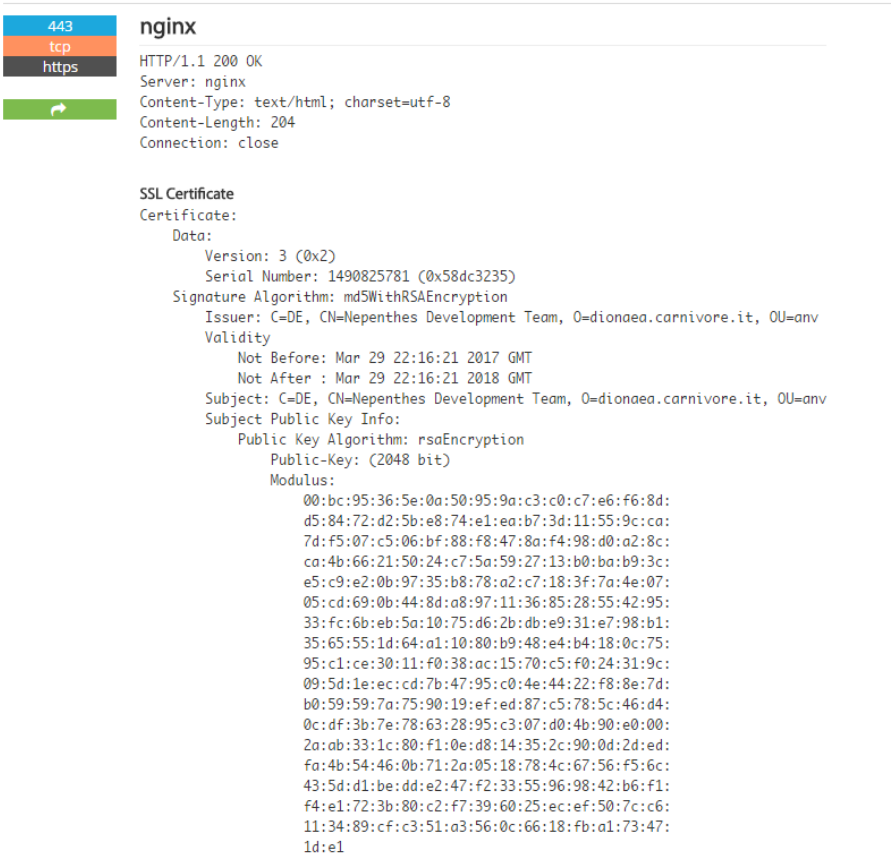


Figura 7-8: Análisis Shodan Dionaea

Se observa en los campos de "Issuer" y "Subject" que se referencia al equipo de desarrollo de Nephentes (el antecesor de Dionaea) y el dominio donde el honeypot Dionaea estaba antes alojado. Se ha cambiado dicho certificado accediendo a la carpeta "/opt/Dionaea/src/connection.c" y cambiado los parámetros:

```
X509_NAME_add_entry_by_txt(name,"CN", MBSTRING_ASC, (const unsigned char
*)"Nepenthes Development Team", -1, -1, 0);
X509_NAME_add_entry_by_txt(name,"O",MBSTRING_ASC, (const unsigned char
*)"dionaea.carnivore.it", -1, -1, 0);
```

a

```
X509_NAME_add_entry_by_txt(name,"CN", MBSTRING_ASC, (const unsigned char
*)"Universidad Autónoma de Madrid", -1, -1, 0);
X509_NAME_add_entry_by_txt(name,"O",MBSTRING_ASC, (const unsigned char
*)"uam.es", -1, -1, 0);
```

El certificado se genera en tiempo de compilación, así que es necesario volver a compilar Dionaea para que el cambio se haga efectivo.

Resultados "Honeypot Or Not?"

Al igual que Glastopf, Dionaea supera con éxito el test.

Resultados FOCA

Ejecutando FOCA en el dominio asignado a Dionaea no se obtienen resultados debido a que FOCA sólo busca los documentos indexados en buscadores como Google o Bing. Siendo una página reciente y que tampoco se ha hecho referencia a ella en otros dominios resulta normal que no esté indexada en Google.

D.2 Nivel Fingerprinting

En este apartado analizaremos los servidores donde están desplegados los honeypots con herramientas especializadas como es Nmap.

D.2.1 Dionaea

Resultados Nmap:

Ejecutando Nmap obtenemos la siguiente información del honeypot Dionaea que se encuentra detrás del Firewall:

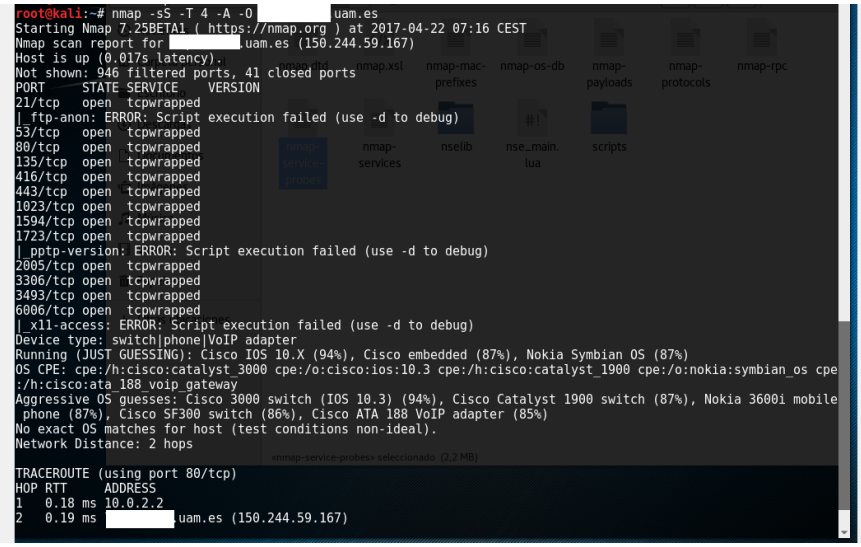


Figura 7-9: Escaneo externo con Nmap a Dionaea detrás del Firewall

Como se ve en la imagen los puertos están siendo filtrados por el Firewall de la UAM con un TCP Wrapper [65]. Esto permite que el escaneo por Nmap no ofrezca información sobre el proceso que se ejecuta en cada puerto. Sin embargo, ello no impide que un usuario se conecte al puerto si sabe qué proceso se ejecuta ahí.

El problema de este análisis es que no vemos si Nmap detecta o no los puertos como un servicio de Dionaea. El siguiente paso será conocer las expresiones que usa Nmap para

detectar los servicios de Dionaea. Si miramos el fichero "probes" de Nmap obtenemos la siguiente información:

Figura 7-10: Reglas de Nmap para la detección de Dionaea

Como se observa en las reglas que utiliza Nmap, a cada servicio que detecta del servidor le aplica una regla. El objetivo es cambiar en Dionaea los parámetros que permiten a Nmap detectar al honeypot.

A continuación, se evaluó cada regla. Los cambios que se han realizado en Dionaea son los siguientes:

Servicio FTP: como se ve en la regla de Nmap "`match ftp m|^220 Welcome to the ftp service\r\n| p/Dionaea honeypot ftpd/`", lo que de este servicio es el banner inicial que envía Dionaea al realizar una conexión exitosa por FTP.
El cambio que se ha realizado es renombrar dicho banner. En "`/opt/dionaea/lib/dionaea/python/dionaea/ftp.py`", se ha cambiado:

```
"welcome_msg": "220 Welcome to the ftp service".
```

a

```
"welcome_msg": "220 Microsoft FTP server".
```

También se ha cambiado en "`/opt/Dionaea/etc/Dionaea/services-enabled/ftp.yaml`" el mensaje de bienvenida:

"`welcome_msg: 220 DiskStation FTP server ready.`" -> "`welcome_msg: 220 FTP server ready.`"

Servicio Microsoft-ds (SMB/CIFS): el SMB (*Server Message Block*), es un protocolo de red que permite compartir archivos, impresoras, y otros dispositivos entre nodos de una red de computadoras que usan el sistema operativo Microsoft Windows, luego Microsoft lo renombró a CIFS (*Common Internet File System*). Para este caso Nmap usa una regla un poco más compleja:

"`\x00\x34\0W\0O\0R\0K\0G\0R\0O\0U\0P\0\0\0H\0O\0M\0E\0U\0S\0E\0R\0-\0.\0.\0.\0.\0.\0.`".

Este se corresponde con los campos que representan el nombre del dominio y del equipo, es decir, si el nombre del dominio es WORKGROUP y el nombre del equipo es HOMEUSER-XXXXXX, donde X es cualquier carácter alfanumérico, entonces el servicio es identificado como un servicio de Dionaea.

Cambiando los campos del nombre del dominio y del servidor que se en el archivo "`opt/dionaea/lib/dionaea/python/dionaea/smb/include/smbfields.py`" solventamos esta identificación:

```
OemDomainName: WORKGROUP → LAPITA
ServerName: HOMEUSER-3AF6FE → SERVER-05X
```

Servicio MSSQL: El tercer servicio identificado es el sistema de base de datos Microsoft SQL Server en el puerto 1433. Si buscamos este servicio en el fichero de patrones de Nmap, se puede ver una cadena en hexadecimal:

```
"match ms-sql-s
m|^\x04\x01\x00\x2b\x00\x00\x00\x00\x00\x00\x1a\x00\x06\x01\x00\x20\x00\x0
1\x02\x00\x21\x00\x01\x03\x00\x22\x00\x00\x04\x00\x22\x00\x01\xff\x08\x00\
x02\x10\x00\x00\x02\x00\x00| p/Dionaea honeypot MS-SQL server/"
```

Esta cadena se corresponde con la respuesta del honeypot a un paquete TDS (Tabular Data Streams) de pre-login en el proceso de conexión al servicio de base de datos. Si se realiza una captura de paquetes con un sniffer y analizamos las conexiones, podemos ver como esta cadena es la correspondiente al campo Token Type (en codificación hexadecimal) que tiene el valor 0x00 (el tercer parámetro).

Luego podemos cambiar dicho valor (por alguno soportado por el protocolo TDS) en el archivo "`opt/dionaea/lib/dionaea/python/dionaea/mssql/mssql.py`". En este caso se ha decidido cambiar el valor a 0xFE.

```
r.VersionToken.TokenType = 0x00 → 0xFE.
```

El código 0xFE corresponde a "Process Done" pero se podría haber escogido cualquier otro. De esta forma hemos cambiado uno de los parámetros que detecta Nmap como honeypot.

Servicio HTTP: si se analiza la regla que utiliza Nmap para el servicio HTTP:

```
"match http m|^HTTP/1\.0 200 OK\r\nContent-type: text/html; charset=utf-
8\r\nContent-Length: 204\r\n\r\n<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML
3\.2 Final//EN\"><html>\n<title>Directory listing for
/</title>\n<body>\n<h2>Directory listing for /</h2>\n<hr>\n<ul>\n<li><a
href=\"\.\./\">\.\./</a>\n</ul>\n<hr>\n</body>\n</html>\n$| p/Dionaea
honeypot httpd/"
```

Se puede observar que Nmap mira la cabecera del documento, la estructura y la longitud de este. Para evitar la detección de Dionaea por HTTP se han tomado dos medidas:

- Añadir documentos a la carpeta "/opt/dionaea/var/dionaea/roots/www" para evitar la detección por el tamaño.
- Cambiar en "/opt/dionaea/lib/dionaea/python/dionaea/http.py" la cabecera y la estructura del documento html.

Para cambiar la cabecera del documento se ha cambiado:

```
r.append('<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">')
r.append("<html>\n<title>Directory listing for %s</title>\n" %
displaypath)
r.append("<body>\n<h2>Directory listing for %s</h2>\n" % displaypath)
r.append("<hr>\n<ul>\n")
r.append('<li><a href="../">../</a>\n')
```

a

```
r.append('<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
')
r.append("<html>\n<title>Index of %s</title>\n" % displaypath)
r.append("<body>\n<h2>Index of %s</h2>\n" % displaypath)
r.append("<hr>\n\n<ul>\n")
r.append('<li><a href="../">../</a>\n')
```

D.2.2 Cowrie y Glastopf

Resultados Nmap:

Para este caso ejecutaremos volvemos a ejecutar Nmap para obtener información sobre el servidor donde están desplegados Glastopf y Cowrie.

El resultado vuelve a ser parecido al caso de Dionaea los puertos vuelven a estar filtrados por TCP Wrapper y aparecen los servicios de Cowrie y Glastopf como filtrados.

Pasamos a ver el fichero "probes" como en Dionaea y resulta que no existen en Nmap reglas que sirvan para identificar a Cowrie o Glastopf.

D.3 Nivel de análisis de vulnerabilidades

En este apartado se analizarán el tipo de vulnerabilidades que tiene cada Honeypot y que en un sistema real podrían ser explotadas con desastrosas consecuencias. En nuestro caso no existe un sistema real, una vez que el atacante deja el sistema, este se reinicia.

D.3.1 Cowrie

En el caso de Cowrie, como simula un servicio SSH, una vez que el atacante ha accedido al sistema tiene tantos privilegios como el usuario que ha usado para entrar en el sistema (o eso cree el atacante). Por ello la mayoría de ataques se deberían enfocar en obtener usuarios con el nivel máximo de privilegios.

El primer cambio realizado es en el fichero "`/cowrie/data/userdb.txt`" donde se ha añadido los usuarios "root" y "admin" con las contraseñas "123456", "root" y "admin". De esta forma permitimos entrar a un mayor número de atacantes para que intenten explotar nuestro sistema.

Una vez dentro de Cowrie vamos a ver qué comandos se pueden ejecutar para comprobar el realismo del Honeypot. Hay algunos comandos básicos para navegar por los archivos como "ls", "cd" o "pwd" que funcionan perfectamente y otros como editores de texto "vim" o "nano" que no están soportados.

Otra de las pruebas que se hicieron es intentar ejecutar el comando "scp" para la transferencia de archivos mediante SSH. Para ello se ha creado un directorio en Cowrie, "`/var/www/`" donde se ha colocado un archivo "index.html" que se puede descargar con "scp" desde la máquina del atacante. A su vez podemos enviar archivos a Cowrie, pero no se almacenan en los directorios que ve el atacante, si no en una capeta donde están todos los archivos subidos a Cowrie.

D.3.2 Glastopf

Las vulnerabilidades que dispone Glastopf son todas de SQL Injection. Este tipo de vulnerabilidades consisten en acceder a través de los campos de los formularios HTML que llaman a ejecuciones PHP, las cuales envían una *query* o consulta a una base de datos SQL.

De esta forma se introducen comandos SQL a través de HTML que no están previstos, como devolver el nombre de las tablas que contiene la base de datos, o datos personales, tarjetas bancarias… e incluso tirar abajo toda la base de datos. Por lo tanto, es bastante común que los atacantes realicen este tipo de ataques, pues una mala configuración puede suponer consecuencias catastróficas.

Uno de los comandos más comunes es introducir en uno de los campos ' " or "1"="1 ' de esta forma se comprobará si la página está mal configurada.

En el caso de Glastopf, si introducimos este comando en el campo de login podemos ver que la página simplemente se recarga, pero en Glastopf se registra el ataque como SQL Injection y de esta forma capturamos ataques generados por bots.

D.3.3 Dionaea

En Dionaea, dependiendo del puerto al que nos conectamos podemos intentar explotar distintos servicios:

HTTP: únicamente se muestra un directorio, se utilizó Blind SQL para obtener alguna información, pero sin ningún resultado, como si fuese una página estática.

SMB: se probó una vulnerabilidad que sufre Print Spooler Service de Microsoft (MS10-061) [66], que permite ejecutar shell code. Usando de nuevo la consola de Metaspolit con los siguientes comandos:

```
msf>  use exploits/windows/smb/ms10_061_spoolss
msf> show targets
msf> set TARGET dionaea.uam.es
msf> show options
msf> exploit
```

Situados desde fuera del Firewall de la UAM no se pudo iniciar sesión y explotar dicha vulnerabilidad pues la conexión fue rechazada.

D.4 Nivel de ganancia acceso y escalamiento de privilegios

Durante esta fase los atacantes configurarían algún tipo de puerta trasera como dejar algún puerto abierto o algún tipo de malware como troyanos para conseguir el control de la máquina de la víctima.

En nuestro caso, al intentar abrir un puerto desde dentro de uno de los honeypots no se ejecuta el comando o no se reconoce y si se intenta dejar algún tipo de malware se almacena en una carpeta de cuarentena y no se permite ejecutar.

E. Instalación y configuración de la máquina virtual distribuible para el despliegue de sensores

En este apartado se muestra la instalación realizada en la máquina virtual que se distribuirá.

Inicialmente se probó a usar Honeydrive [67], una máquina virtual distribuible como el que deseábamos hacer con varios honeypots preinstalados. La instalación de Honeydrive es muy simple, como se demuestra en el anexo A.6. Lamentablemente al intentar actualizar los paquetes de Ubuntu saltaba siempre error. Por ello se optó por crear una máquina virtual distribuible partir de las máquinas que hemos utilizado en nuestra red local.

E.1 Preparación de la máquina virtual

Se ha realizado una instalación igual a la de los honeypots que se han usado en nuestra red local. Esto incluye la instalación de los tres honeypots (Dionaea, Cowrie y Glastopf) y Filebeat que se encargará de enviar los logs creados por los honeypots.

Además, se ha configurado para que el servicio de Filebeat se ejecute cada vez que se inicia la máquina automáticamente. Para ello se ha ejecutado en una terminal:

```
$ sudo update-rc.d filebeat defaults
```

Como es una plantilla, el identificador que se le ha asignado temporalmente (en la configuración de Filebeat) es “Guest”. El usuario al no tener permisos de root no puede cambiar dicho identificador.

E.2 Ejecución automática

Primeramente, creamos un script (llamado “init_honeypots.sh”) para Shell que inicie los honeypots:

```
#! /bin/bash

#Select honeypots to start
glastopf=true
dionaea=true
cowrie=true

#Start Glastopf
if [ "$glastopf" = true ]
then
        echo 'Starting Glastopf'
        sudo /opt/myhoneypot/glastopf-runner
fi

#Start Cowrie
if [ "$cowrie" = true ]
then
```

```
        echo 'Starting Cowrie'
        su -c "/home/cowrie/cowrie/./start.sh" cowrie
fi

#Start Dionaea
if [ "$dionaea" = true ]
then
        echo 'Starting Dionaea'
        sudo service dionaea restart
fi
```

En este script podemos decidir qué honeypots ejecutar o dejar apagados. Por ello se proponen dos opciones:

- Dejar que el usuario ejecute el script y por tanto otorgarle privilegios de root (con muchas consecuencias no deseables).
- Ejecutar el script cada vez que se inicie el honeypot de forma automática con una configuración preestablecida.

En este desplegable se optó por la segunda opción y para ejecutar el script cada vez que se inicia la máquina virtual se realizaron los siguientes pasos:

Copiamos el script a la carpeta "`/etc/`", aunque puede ser cualquiera.
Creamos un script en la carpeta "`/etc/init.d/`" al que llamaremos "`run_honeypots`".
Dentro del script escribimos la carpeta donde se encuentra "`init_honeypots`":

```
#!/bin/sh
/etc/init_honeypots.sh
```

Hacemos que el script sea ejecutable con el comando:

```
$ chmod ugo+x /etc/init.d/run_honeypots
```

Por último, añadimos el script a la lista de inicio:

```
$ update-rc.d run_honeypots defaults
```

Y de esta forma permitimos que el script se ejecute al iniciar la máquina virtual.

Al ejecutar los tres honeypots a la vez sacrificamos el servicio HTTP de Dionaea para que Glastopf pueda desplegarse en el puerto 80. Es preferible al de Dionaea pues Glastopf contiene vulnerabilidades de inyección SQL en su servicio y Dionaea no.

Como la configuración de Filebeat ya está realizada, todos los datos que recoja se enviarán automáticamente al servidor figurando con nombre "Guest" (variará la IP dependiendo del usuario)

Por último, para pasar a los usuarios la máquina virtual, se generó un fichero ".ova" desde Virtualbox [68] , y así pueden desplegar la máquina de una forma cómoda. En un inicio se creó un link en Dropbox para que se pudiese compartir con gente de confianza. Posteriormente se podrá utilizar otro tipo de servidor para que los usuarios puedan descargársela.